Advanced Modeling & Simulation (AMS) Seminar Series
NASA Ames Research Center, January 28th, 2021

Bringing SIMULATION to the 5th DIMENSION…

WARPIV KERNEL

HIGH LEVEL OVERVIEW OF THE PARALLEL AND DISTRIBUTED COMPUTING CAPABILITIES OF THE WARPIV KERNEL

NASA PHASE I SBIR EFFORT
TECHNOLOGIES FOR LARGE-SCALE NUMERICAL SIMULATION

JEFFREY S. STEINMAN, PH.D.
PRESIDENT & CEO, WARPIV TECHNOLOGIES, INC.
JEFFREY.STEINMAN@WARPIV.COM,
(858) 531-0643

BACKGROUND

- Much of this work began at JPL/Caltech from 1988-1996
  - Hypercube project supporting the Strategic Defense Initiative (Star Wars – now Missile Defense)
  - Optimistic parallel discrete-event simulation (TWOS, SPEEDES, 5 NASA patents, 80 papers, etc.)
- The work moved to industry in 1996 where SPEEDES was used on numerous mainstream simulation programs for the DoD
  - Wargame 2000, Joint Modeling And Simulation System (JMASS), Joint Simulation System (JSIMS), Extended Air Defense Test Bed (EADTB), etc.
  - But the multicore computing revolution stalled (federations dominated)… UNTIL NOW!
- WarpIV Kernel began in 2001 as the replacement for SPEEDES and is being used by numerous DoD simulation programs
  - ITASE (DIA), SAFE-SiM (DARPA), MDEM (MDA), AFSIM (Air Force), and growing…
- Goals of this NASA Phase I SBIR Effort
  - Bring new technologies to NASA for large-scale numerical simulations involving both (1) space science and (2) space missions with a secondary goal of integrating these domains
  - Provide technologies and tools that will attract new HPC users (e.g., build simulations on multicore laptops and desktops, and then seamlessly move to HPC platforms to scale up when needed)

SCIENCE & TECHNOLOGY

- Science
  - Model developers, verification and validation of models, users of models, and analysts who study the results of simulation outputs
- Technology
  - Developers of high-speed communication services, simulation engines, modeling frameworks, software utilities, graphical displays, data logging, playback, and cataloging systems to support data mining, etc.
- Analogy – Astronomers (scientists) and Telescope makers (technologists)…
  - Breakthroughs in technology always enables better science…
- R&D is performed in both Science & Technology
  - We want to work with you on HPC technology and tool development for the WarpIV Kernel
  - We want to work with you on space science and mission modeling efforts using our technology

AGENDA

- The WarpIV Kernel Open-Source Code Base
- Communications
  - High Speed Communications, Client/Server ORB, Grid Computing, Optimization Engine, Parallel Application Launching
- Optimistic/Conservative Parallel Discrete-Event Simulation (PDES) Engine with Flow Control
- Modeling Framework Dramatically Reduces Lines of Code for Building Complex Models
  - Event Types, Process Model, Composable Models, Model Editor with Code Generation, Scenario Specification, Publish and subscribe data distribution, Interest Management with Hierarchical Grids, Cognitive Modeling Constructs
- Large Assortment of Software Utilities
- Federations and the High-Level Architecture (HLA)
  - High-Performance Computing Run-Time Infrastructure (HPC-RTI), Local RTI Component Proxy (LrcProxy) Framework, External Modeling Framework (EMF) – Can be used to parallelize a legacy simulation by self-federating
- Bit Compressed Parallel and Distributed Data Logging
- Performance Tools
  - Critical Path Analysis, Software Profiling, Built-In Event-Processing Instrumentation, Communication Latencies
- Cross-Platform Widget Set for Building GUIs, 3D/2D Visualization, and Analysis
- Work performed on this Phase I NASA SBIR Effort
  - Discrete-event vs. time stepping for n-body gravitational problems
  - Planetary Rings, Space Debris in a Missile Defense Scenario, Ellipsoidal Gravity, RF Propagation

THE WARPIV KERNEL CODE BASE

- Supports a large variety of parallel and distributed computing applications
  - Emphasis on modeling & simulation
- More than 825,000 lines of (mostly) C++ code
  - Cross-platform (Linux, Mac OS X, other versions of UNIX, Windows)
  - Fully self-contained with no 3rd party dependencies
  - Easy to build and supports all mainstream compilers
  - Builds into a single library with all header files organized in one directory
- Free and open source with some restrictions on distribution
  - The United States government has an unrestricted license to the WarpIV Kernel code base
  - No classified data or algorithms in the software distribution (no ITAR restrictions)
  - Categorized as EAR-99 by the Department of Commerce (no export license required)
  - No proprietary data-rights claims or patents on the WarpIV Kernel software
  - WarpIV Technologies, Inc. maintains distribution rights for commercial use

LAYERED ARCHITECTURE

Composable Application *Physics, Engineering, Engagement, Mission, Campaign*	*Legacy Simulation Engine & Models*	*Legacy HLA Federate*	*LVC – Federation & Enterprises*	*External Systems, Visualization, & Analysis*
	Middleware & Code Generation (Engine Bridge)	HPC-RTI & LrcProxy (HLA Bridge)	LVC & Web-based Gateways	External Modeling Framework (EMF)
Composable Systems				
Composable Model & Service Packages (MDEM)		Cloud Services, PAL, Grid Computing, Optimization Engine		External Comm.
SOM/FOM Data Translation Services to Support Integration with Arbitrary Object Models				
Cognitive Architecture Framework (Thought Triggering, Mission Goals, Intelligent Behavior)				
Distributed Simulation Management Services (Pub/Sub Data Distribution for Objects and Interactions)				
Standard Modeling Framework to Promote Composability and Rapid Model Development				
Optimistic and Conservative Time Management				
Event Management Services for Supporting the SimEngine				
Standard Template Library (STL) & Object-Oriented Persistence				
Parallel and Distributed Data Logging Services				
Internal High-Speed Communications				
ORB Client/Server Network Services				
Rollback Utilities				
Rollback Framework				
General Purpose Software Utilities				
Threads and Inter-Process Communication Abstractions				
Operating System Services				

Legend:

- OS/IPC Services
- User APIs
- Framework
- LVC
- Middleware
- *Application Code*

New DOC Ruling: EAR-99
International distribution
No export license required

HIGH SPEED COMMUNICATIONS

COMMUNICATIONS

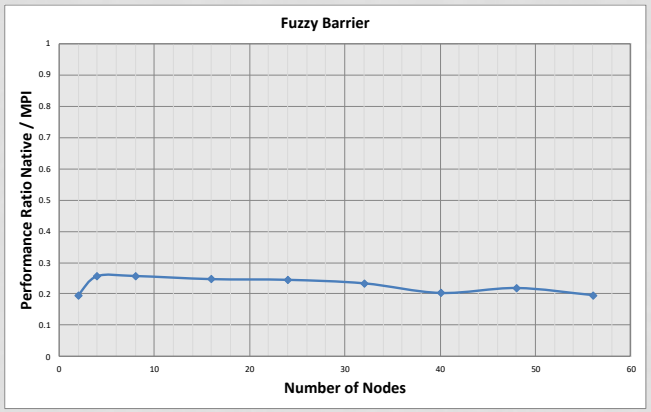
- High Speed Communications
  - Better performance than MPI and much easier to use
  - 32M short messages per second on a NUMA 32-core AMD Ryzen Threadripper
- Client/Server Object Request Broker (ORB)
  - Provides reliable network communications (TCP/IP) between clients and servers
  - High-level abstraction to invoke remote methods on server objects via defined interfaces
  - Supports asynchronous, functions, and two-way services
- Grid Computing
  - Tasker farms work to Workers with built-in fault tolerance
- Optimization Engine
  - Builds on the Grid-Computing framework to intelligently guide concurrently executing simulations towards a goal (optimization, Nash Equilibriums, formation of response surfaces, etc.)
- Parallel Application Launching (PAL)
  - Launches homogeneous or heterogeneous applications on one or more machines in a robust and repeatable manner

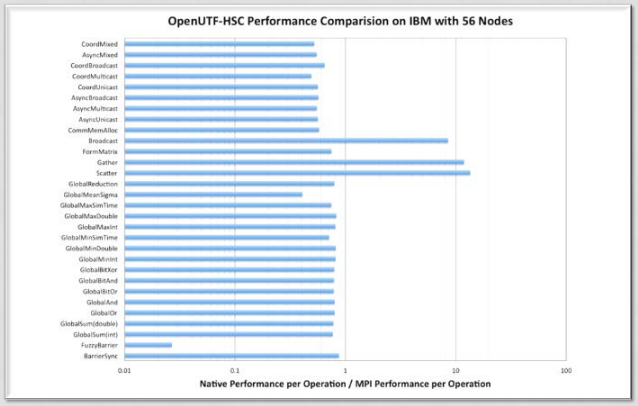
HIGH SPEED COMMUNICATION SERVICES

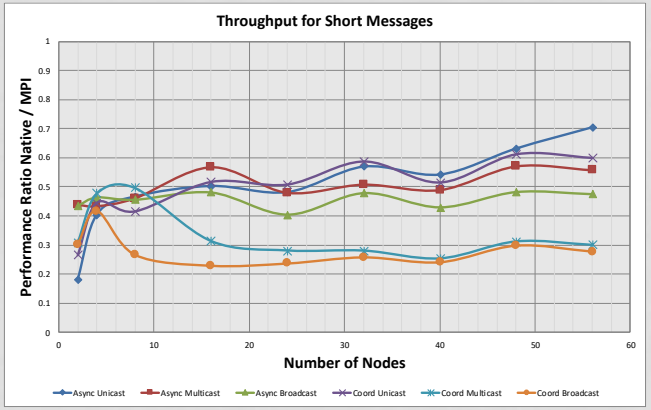
- Startup and Terminate
  - Forks/spawns local processes
  - Manages shared memory and network connections
- Synchronized Wall Clock Services
  - Normally performed at startup
  - The wall time is set to 0 at startup
- Miscellaneous services
  - Node info, shared memory tuning parameters, etc.
- Synchronization
  - Hard and fuzzy barriers
- Global reductions
  - Min, Max, Sum, Product, etc.
  - Mean and Standard Deviation for Node Statistics
  - Can support user-defined operations
- Synchronized data distribution
  - Broadcast, Scatter, Gather, Form Vector/Matrix

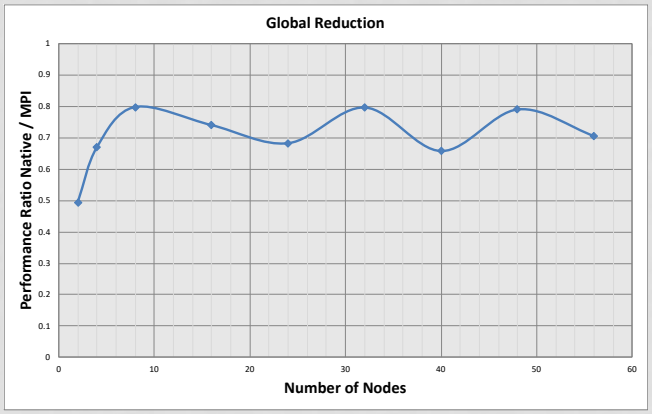
- Asynchronous Message Passing
  - Unicast, destination-based multicast, broadcast
  - Up to 512 message types
  - Automatic or user-defined memory allocation
- Coordinated Message Passing
  - Patterned after the Hypercube Crystal Router
  - Synchronized operation guarantees that all messages are received by all nodes
  - Unicast, destination-based multicast, broadcast
  - Automatic or user-defined memory allocation
- ORB Services
  - Remote asynchronous method invocation with user-specified interfaces

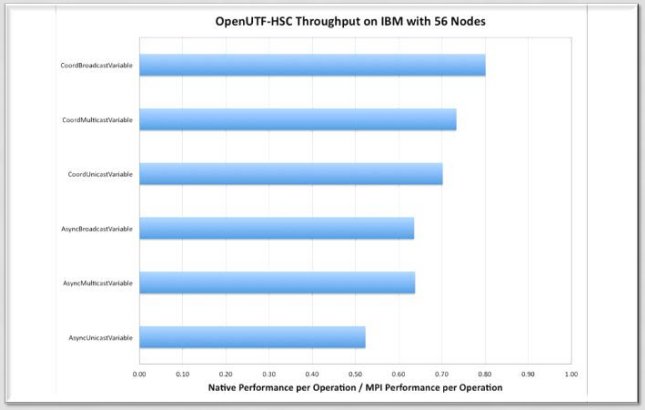
WARPIV HSC VS. MPI BENCHMARKS

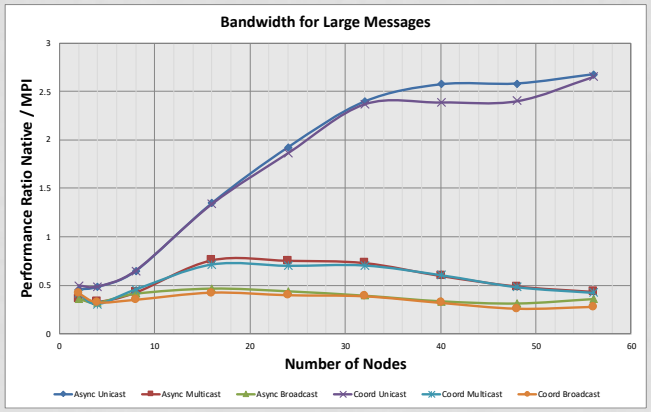






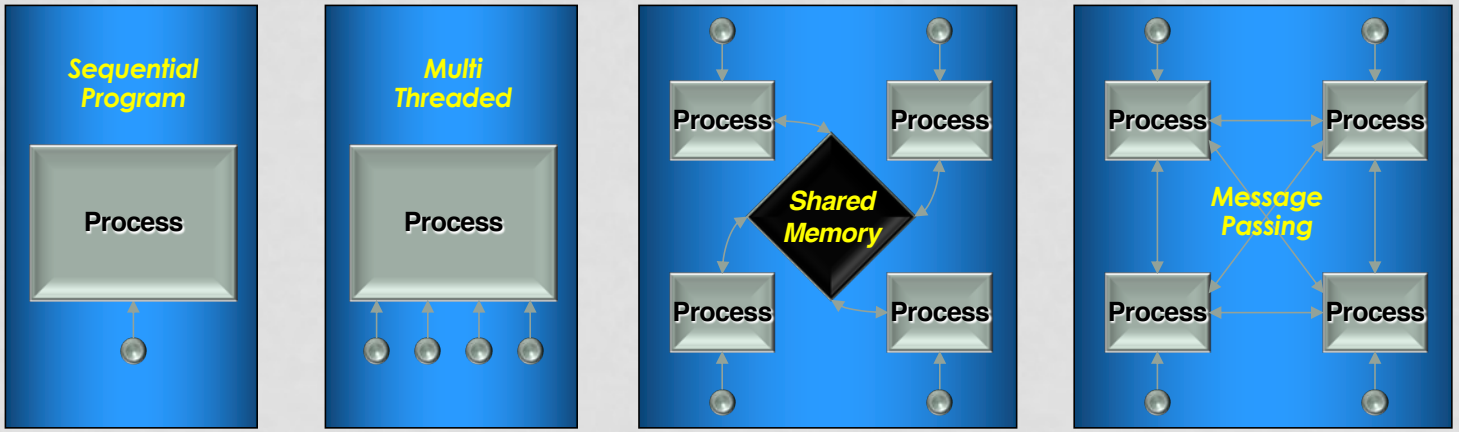




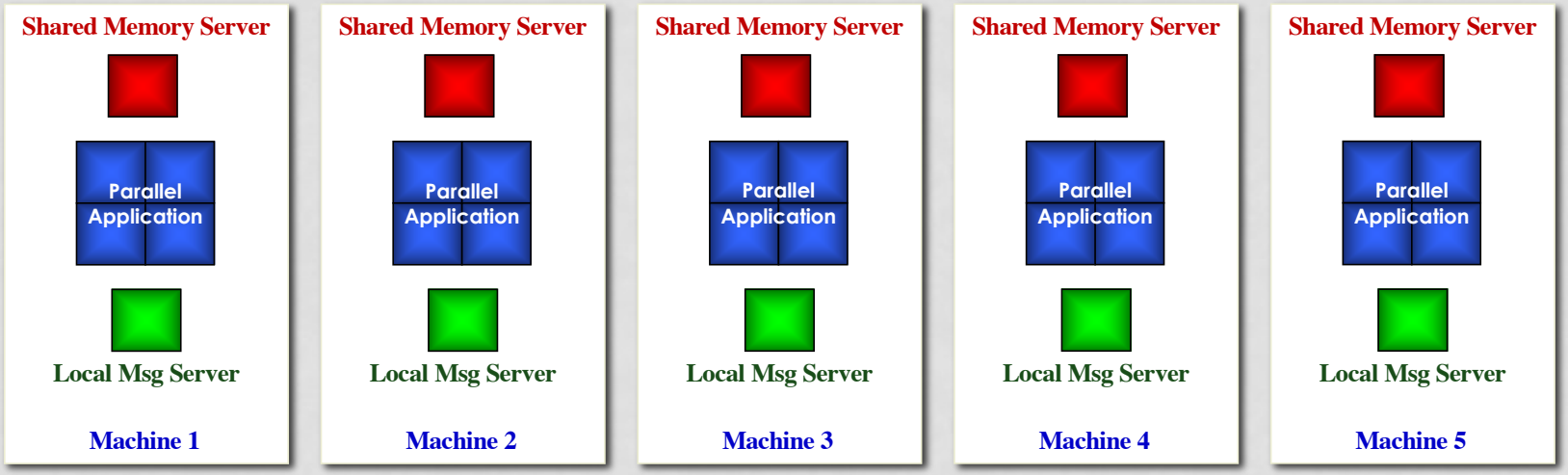


APPROACHES TO PARALLELISM

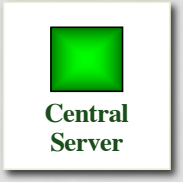
- Distributed net-centric computing
  - Programs communicate through a network interface
    - TCP/IP, UDP/IP, Multicast, HTTPS, SOA and Web Services, Client/Server, CORBA, Federations, Enterprises, Grid Computing, etc.
- Parallel multicore computing on a single machine
  - Processors directly communicate through high speed mechanisms
    - Threads, shared memory, reliable message passing



SHARED MEMORY AND DISTRIBUTED COMPUTING



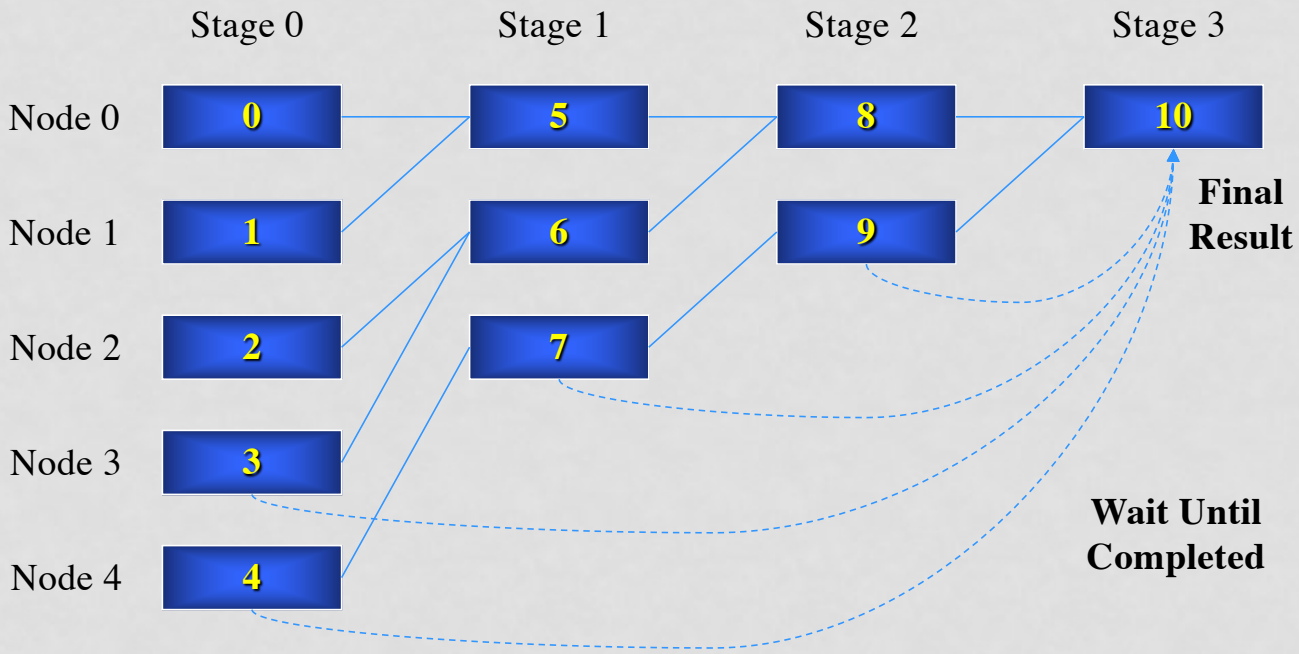
- **Shared Memory Server** creates and deletes shared memory segments for the local nodes running the parallel application when the high-speed comm starts up and later terminates. It kills all nodes on the machine if a node crashes (i.e., disconnects unexpectedly).
- All nodes on a machine send their remote messages first to their **Local Msg Server** (low overhead), which then forwards the message across the network to its destination. If more than one remote node needs the message (multicast or broadcast), it is sent once to the machine and then disseminated to the other receiving nodes through shared memory.



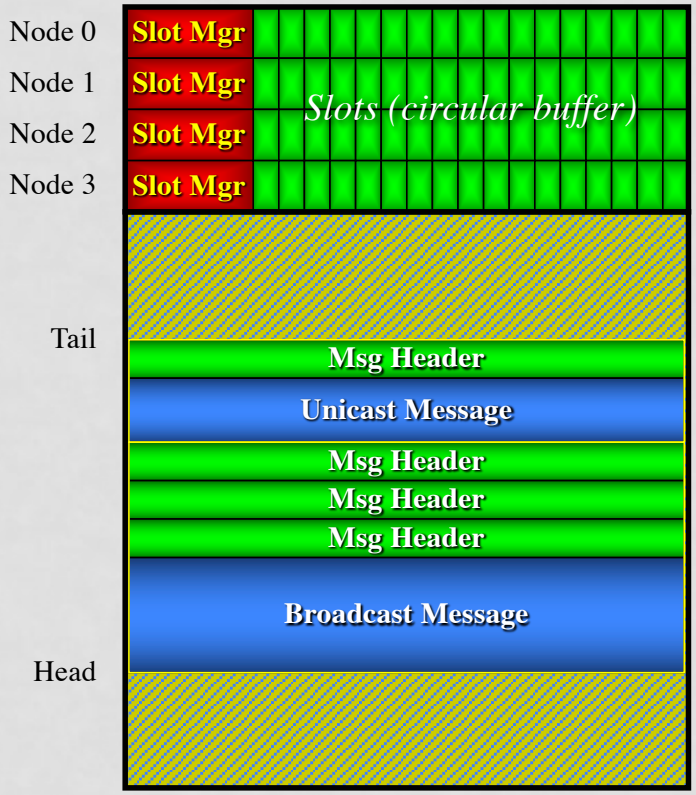
- **Central Server** handles barrier synchronizations and global reductions. If a node unexpectedly disconnects, it sends a kill message to local node 0 on all other machines, which causes the node to exit, which then causes its shared memory server to kill all the other nodes.

GLOBAL SYNCHRONIZATIONS AND REDUCTIONS

Example of a global synchronization on five processing nodes



MESSAGE SHARED MEMORY STRUCTURE



Output Messages (circular buffer)

One shared memory block per node

- Slots manage incoming messages for each node
- Circular buffer manages outgoing messages

Steps in sending a message:

1. Write header and message to head in senders output message buffer.
2. Write index of msg header in the receiving node shared memory slot for the senders node.

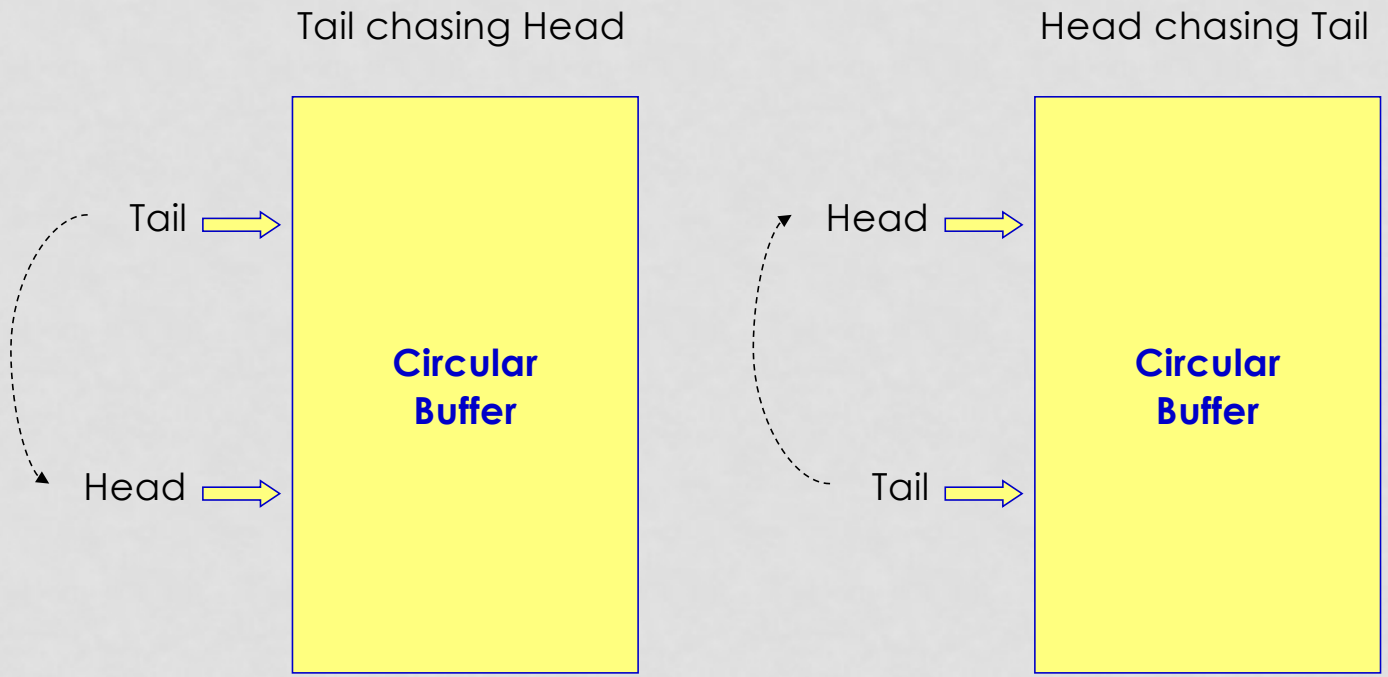
Steps in receiving a message

1. Iterate over slot mgrs to find messages
2. Read message using index in the slot
3. Mark the header as being read

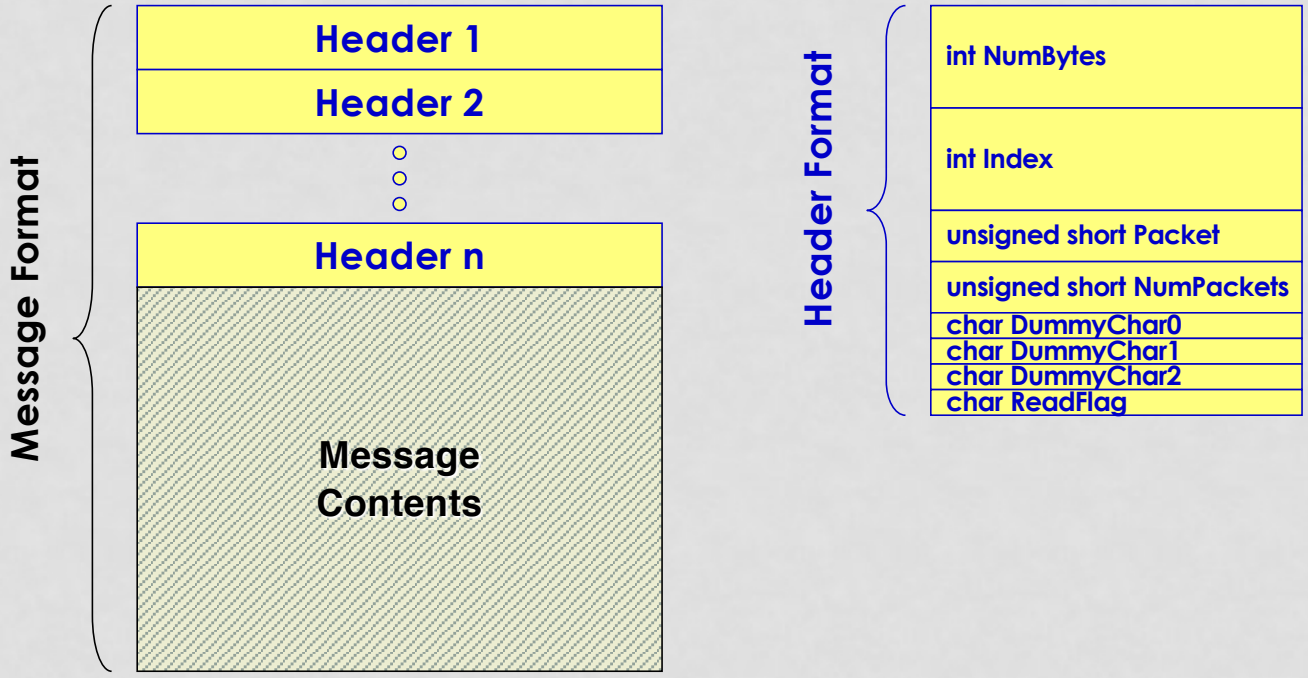
Potential technical issues

- Cache coherency
- Instruction synchronization

CIRCULAR BUFFER HEAD AND TAIL INDICES



MESSAGE AND HEADER FORMATS



OPTIMISTIC PARALLEL DISCRETE-EVENT SIMULATION (PDES) ENGINE WITH FLOW CONTROL

TIME STEPPING VS. DISCRETE EVENT

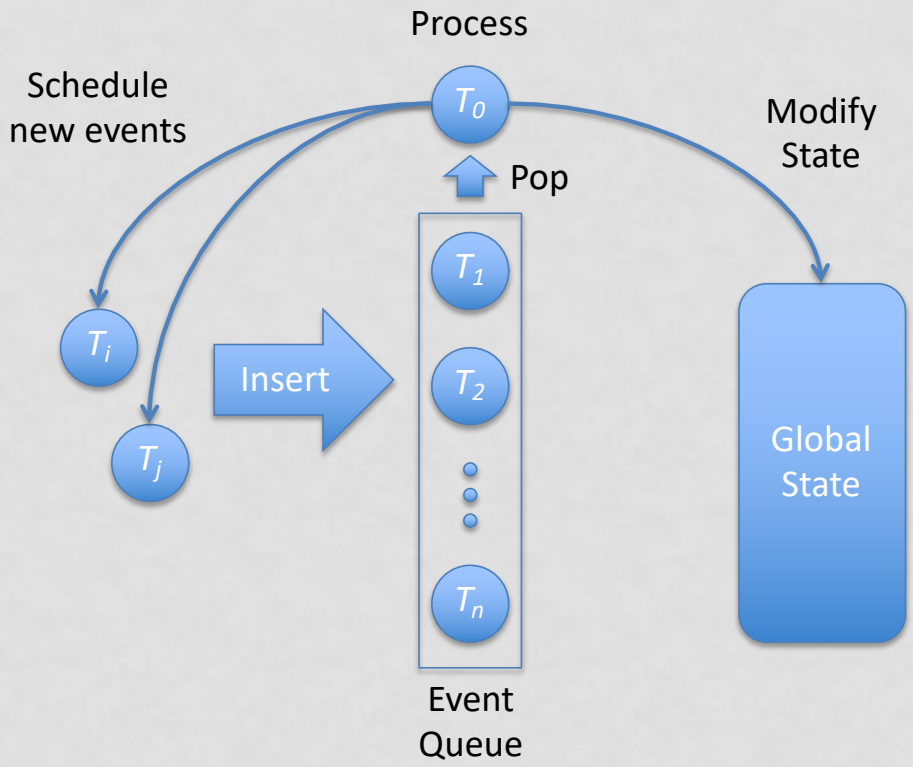
Parallel Time Stepping

```
InitializeState();

double endTime = 100.0;
double step = 1.0;

for (double time=0.0; time<=endTime; time+=step) {
  Compute();
  Communicate();
  …
}
```

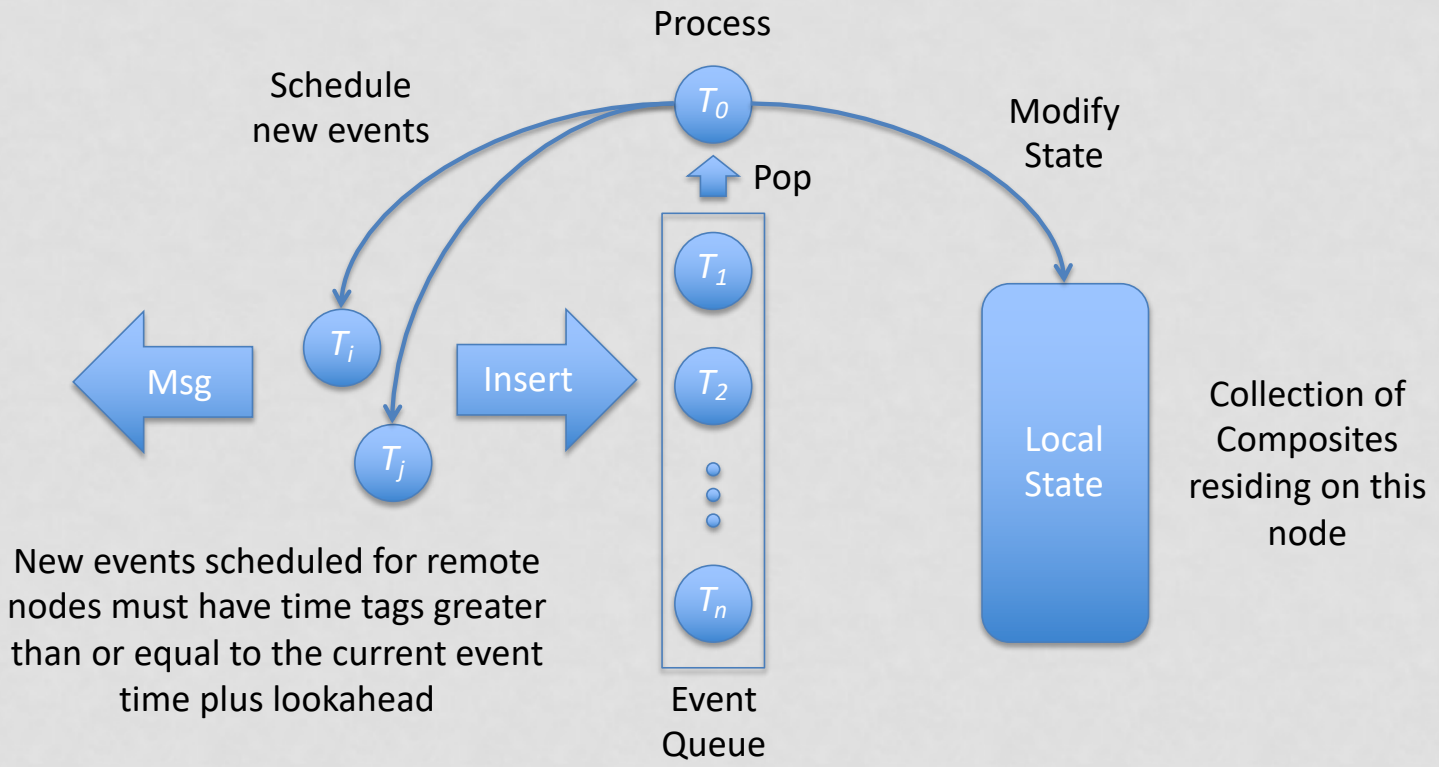
Sequential Discrete Event Simulation

```
InitializeState();
ScheduleInitialEvents();

double endTime = 100.0;

while (eventQueue->GetNextEventTime()<=endTime) {
  double time = EventQueue->GetNextEventTime();
  Event *event = EventQueue->PopNextEvent();
  event->Process(time);
  event->InsertNewEvents(EventQueue);
}
```

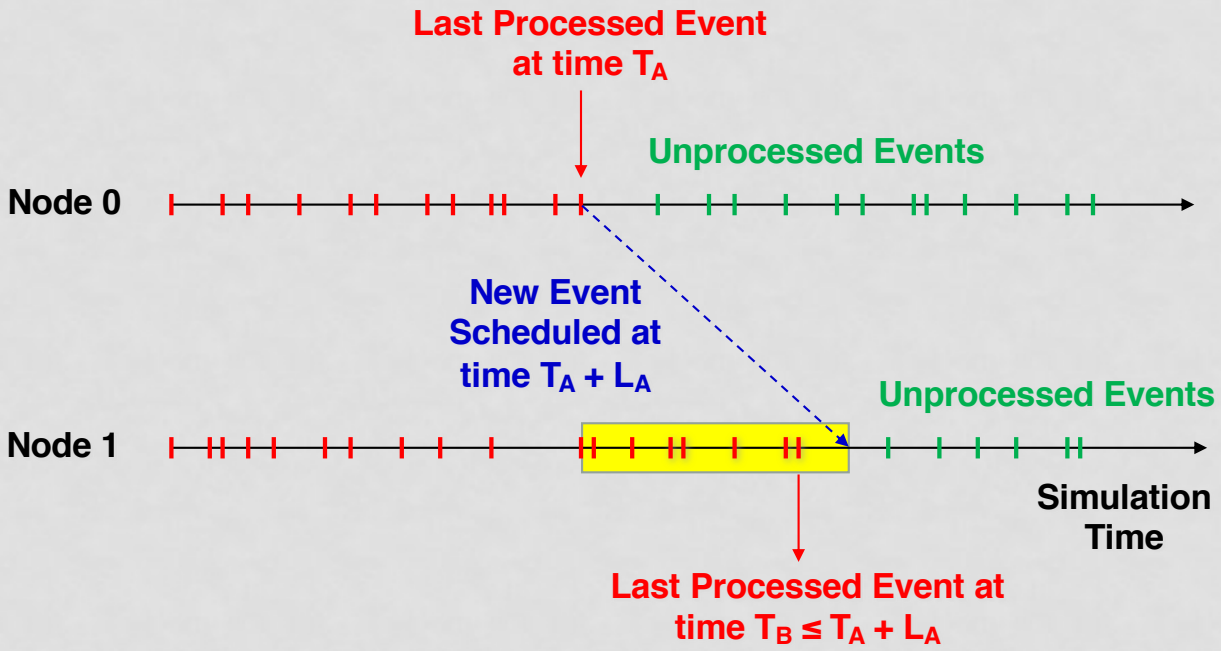
SEQUENTIAL DISCRETE EVENT SIMULATION



CONSERVATIVE PARALLEL DISCRETE EVENT SIMULATION

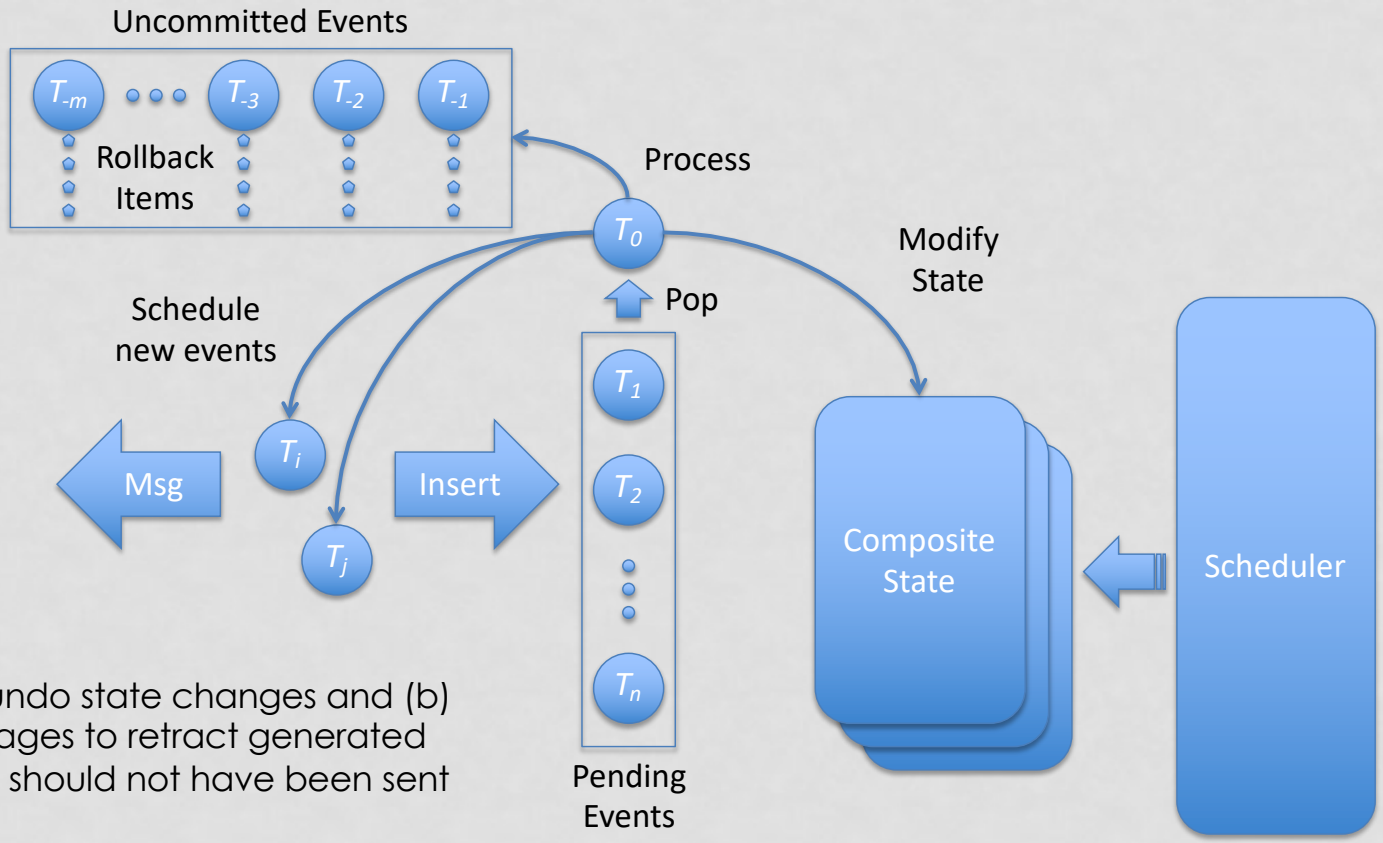


LOOKAHEAD & CONSERVATIVE METHODS



Lookahead: Promise between nodes to never schedule remote events with time tags less than their current time + Lookahead

OPTIMISTIC PARALLEL DISCRETE EVENT SIMULATION

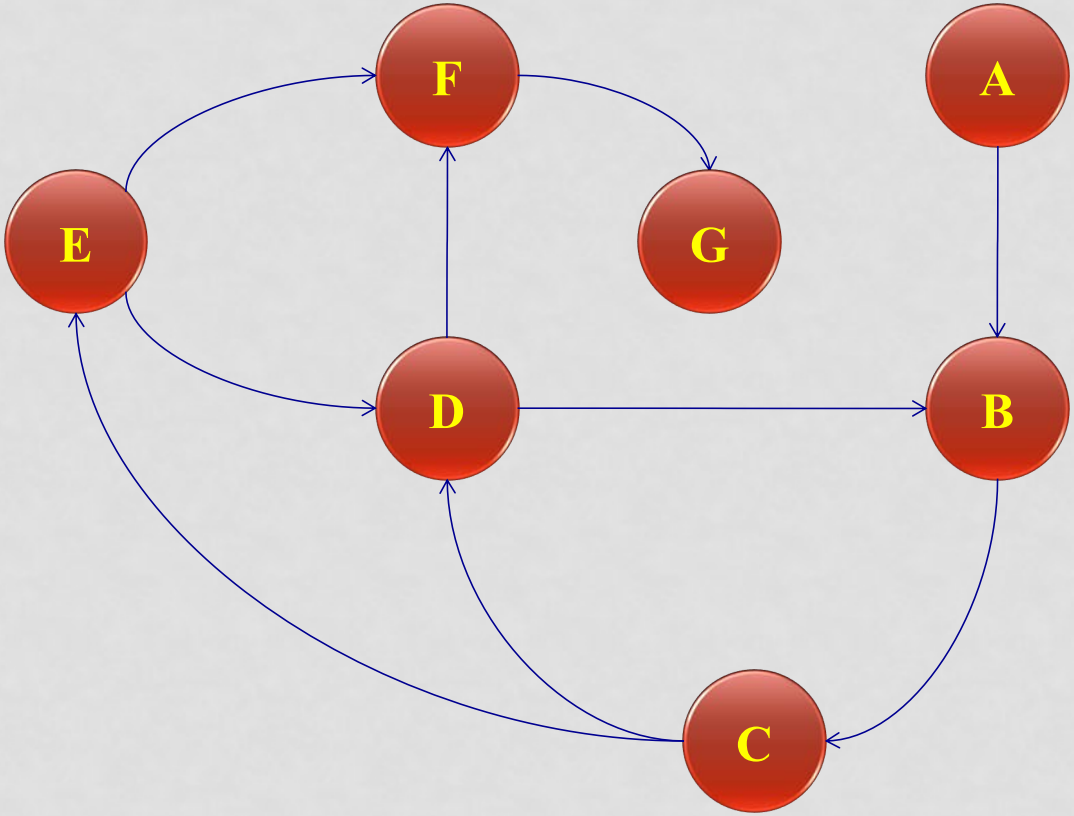


Rollbacks (a) undo state changes and (b) send antimessages to retract generated messages that should not have been sent

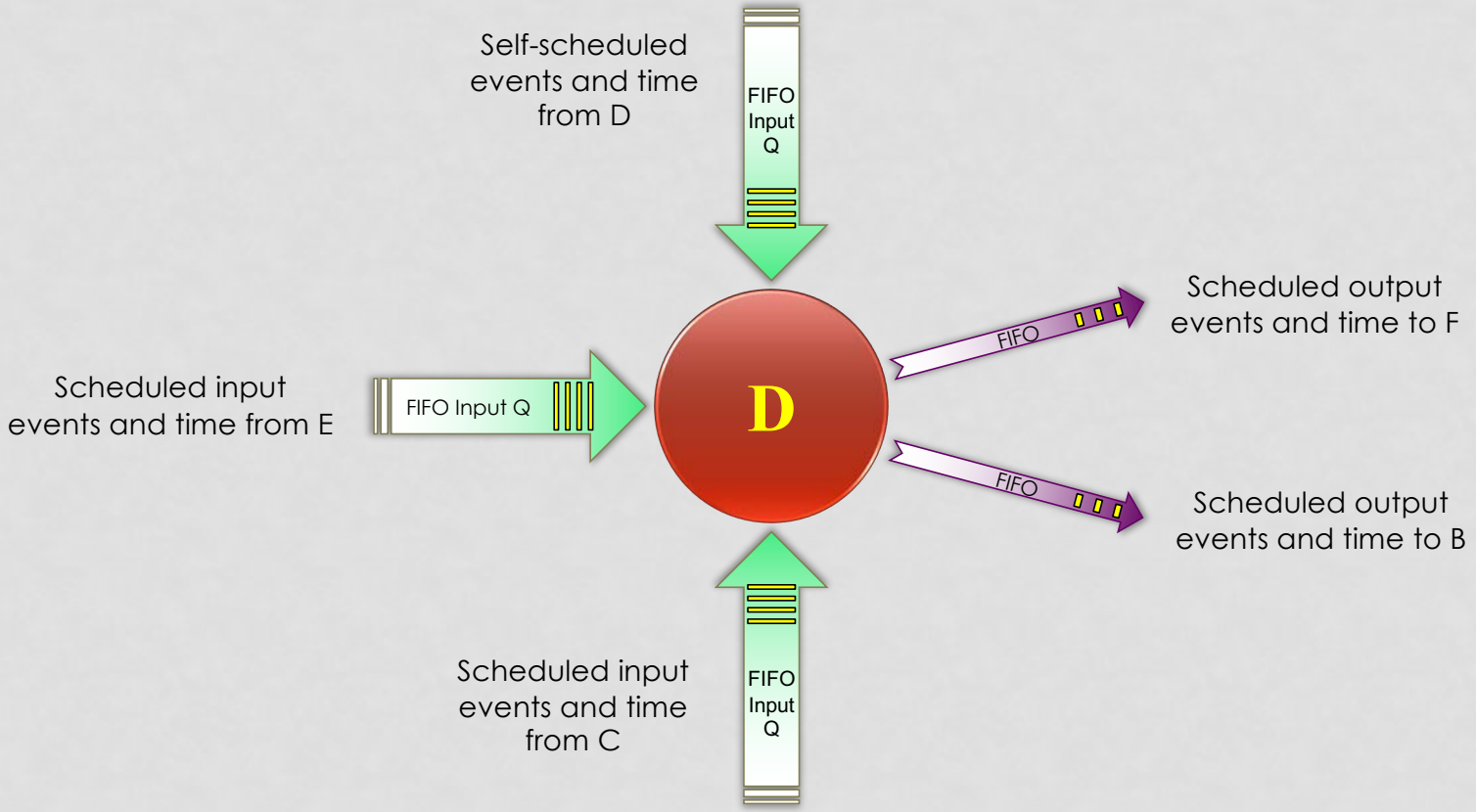
CONSERVATIVE VS OPTIMISTIC METHODS

- Conservative algorithms impose one or more constraints
  - Object interactions limited to just "neighbors" (e.g., Chandy-Misra)
  - Object interactions must have artificial delays (e.g., lookahead)
  - Object interactions follow FIFO constraint
- Optimistic algorithms impose no constraints but require a more sophisticated engine
  - Support for rollbacks (and advanced features for rollforward)
  - Require flow control to provide stability
  - Optimistic approaches can support real-time applications better than conservative approaches...
- The most important thing is for applications to develop their models to maximize parallelism
  - Simulations will not execute in parallel faster than their critical path

TOPOLOGY BASED SYNCHRONIZATION



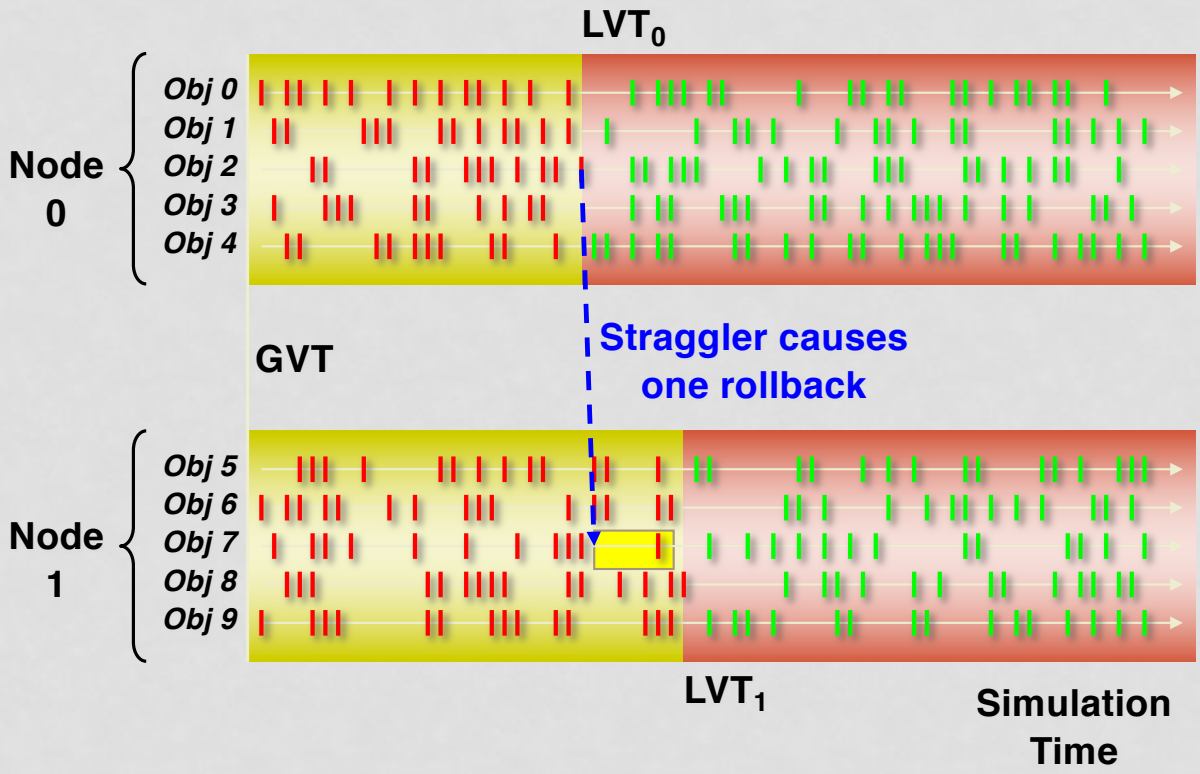
FIFO CONSTRAINT



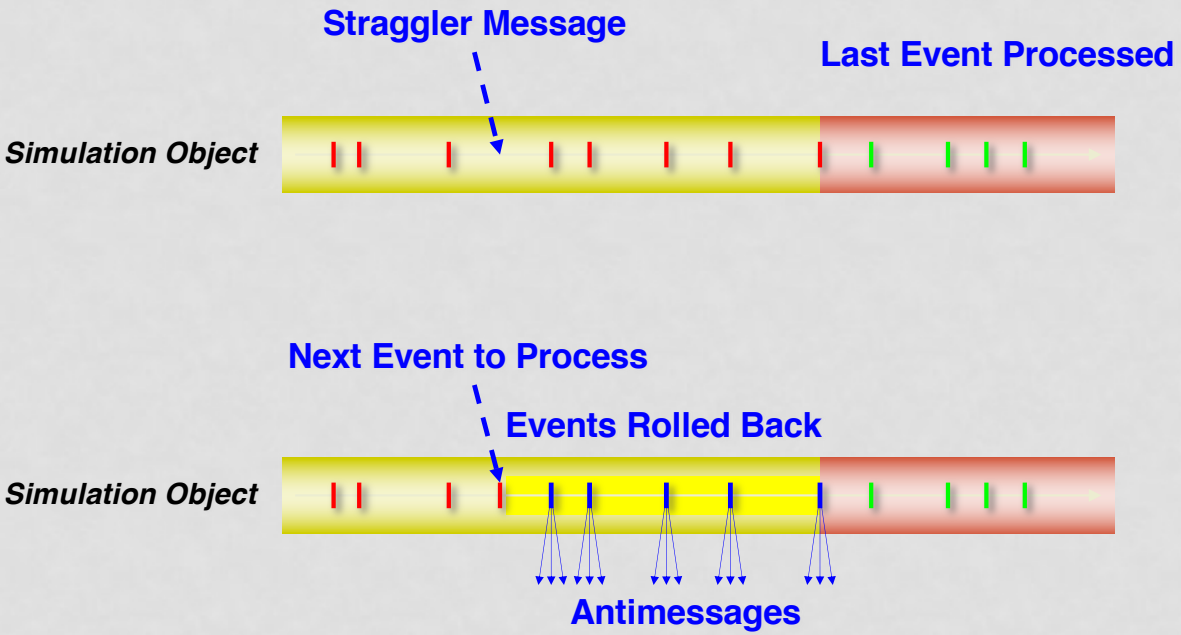
OPTIMISTIC SIMULATION AND GLOBAL VIRTUAL TIME

- Global Virtual Time (GVT) is a concept that defines the progress of the parallel simulation. It is defined as the minimum time-tag of:
  - Unprocessed (pending) event
  - Unsent message
  - Message or antimessage in transit
- Theoretically, GVT changes as events are processed
  - In practice, GVT is updated periodically by a GVT update algorithm
- To correctly provide time management services to the outside world, GVT must be updated synchronously between internal nodes
  - Frequent GVT updates supports tight interactions, but can also introduce overheads
- Processed events with time tags less than GVT are committed once GVT is updated
  - Memory to capture rollbacks (e.g., state changes, messages, I/O, etc.) can be deallocated

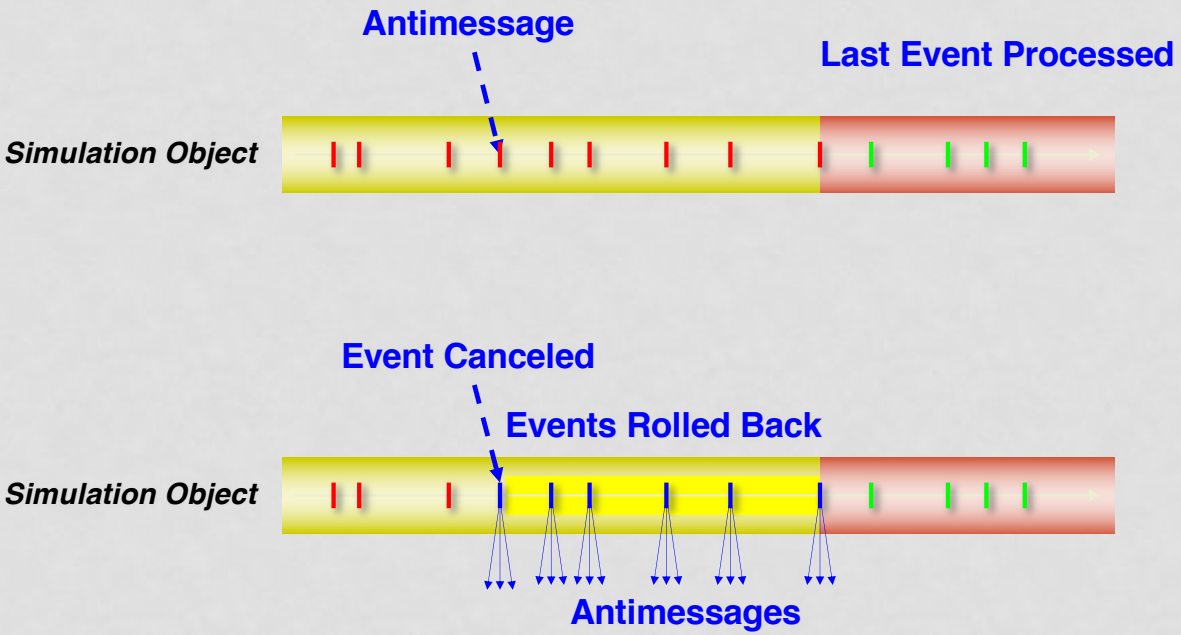
ROLLBACKS OCCUR WITH OBJECTS (NOT BY NODES)



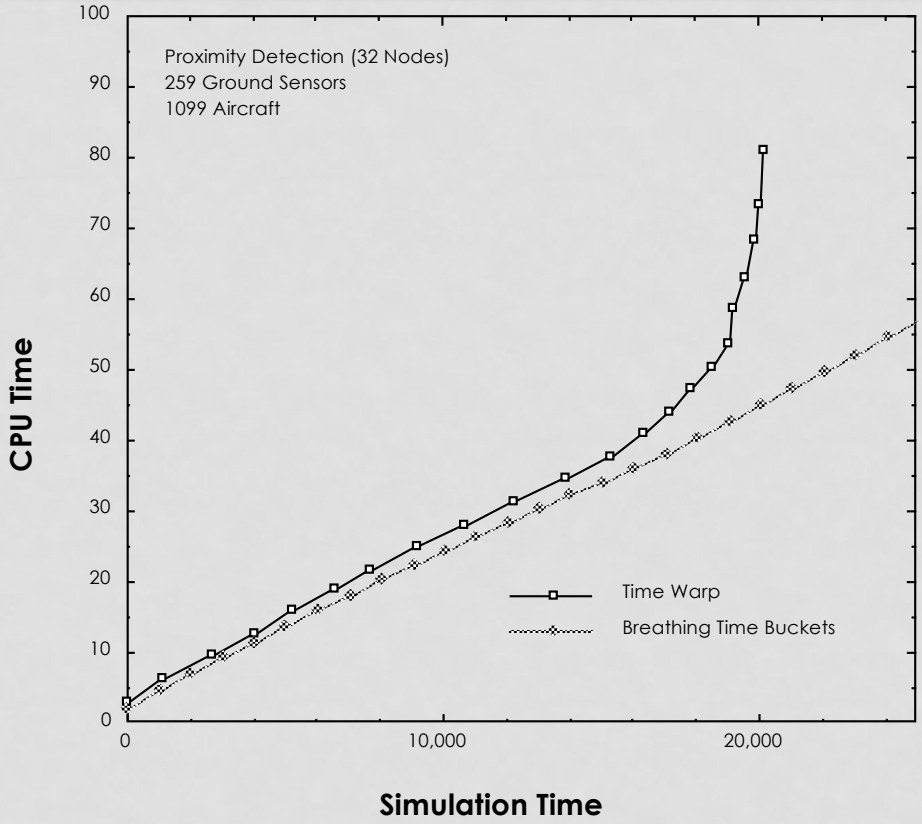
TIME WARP & STRAGGLER MESSAGES



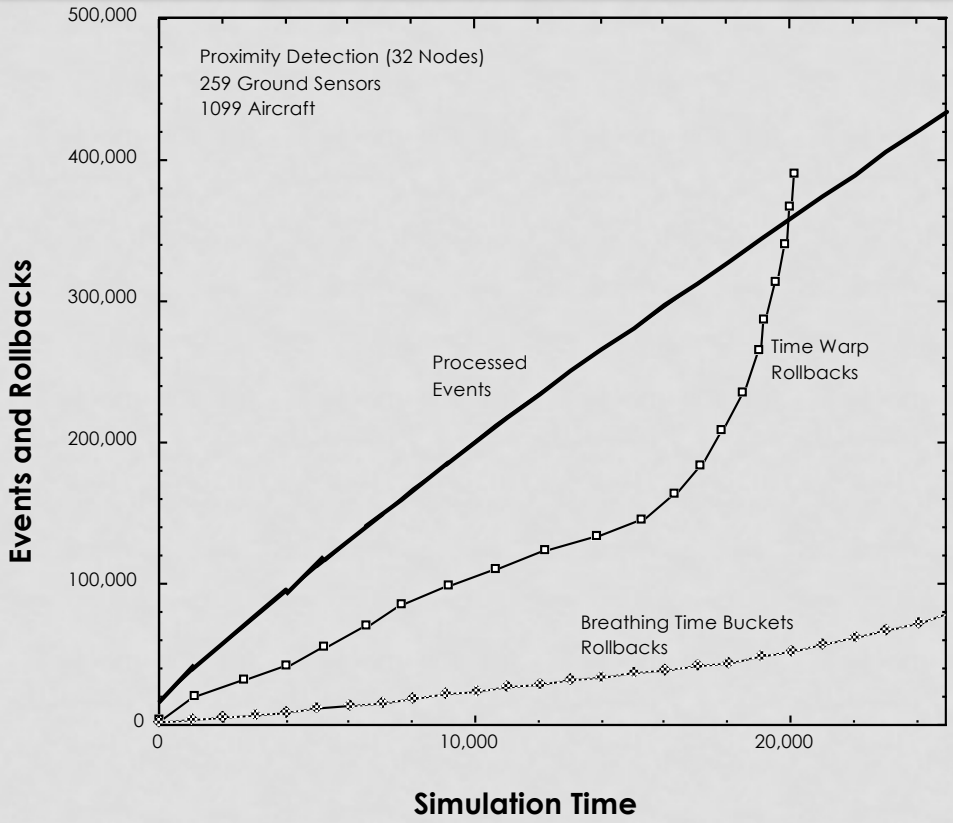
TIME WARP & ANTIMESSAGES



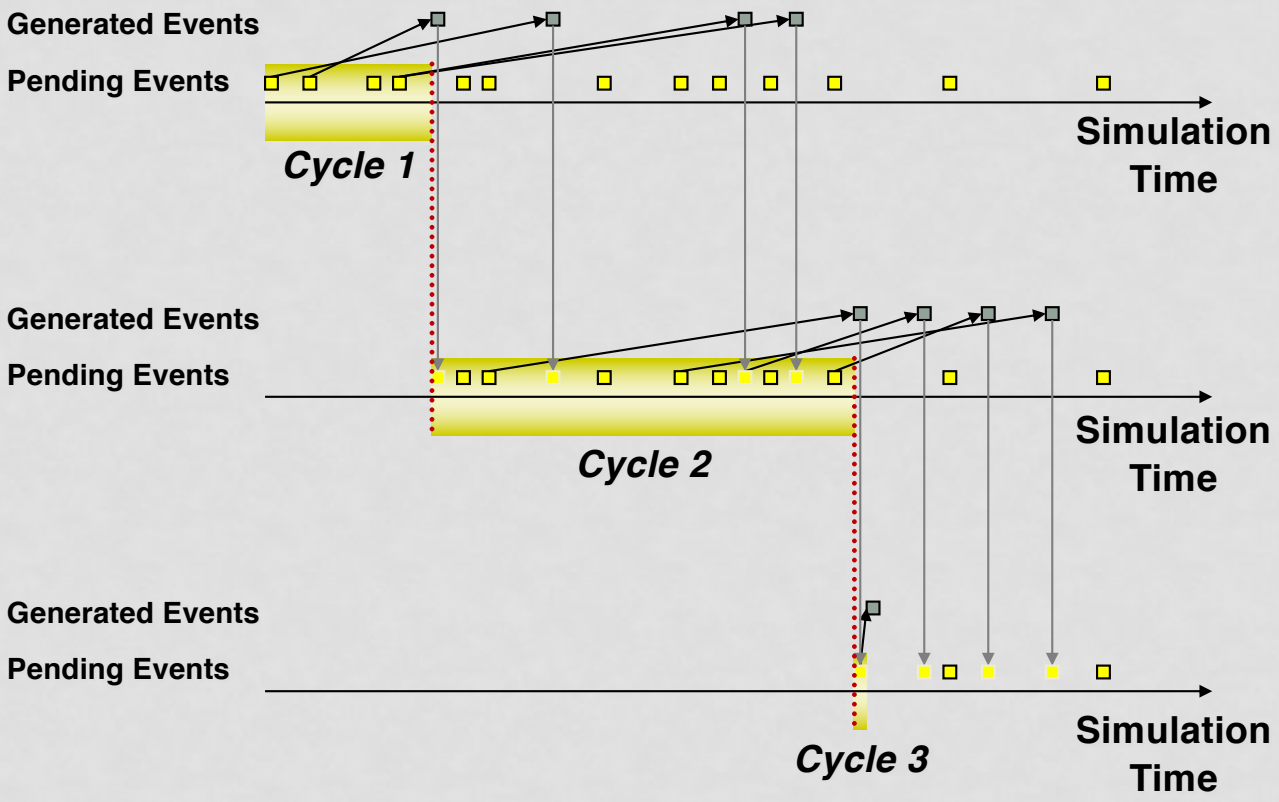
TIME WARP INSTABILITIES



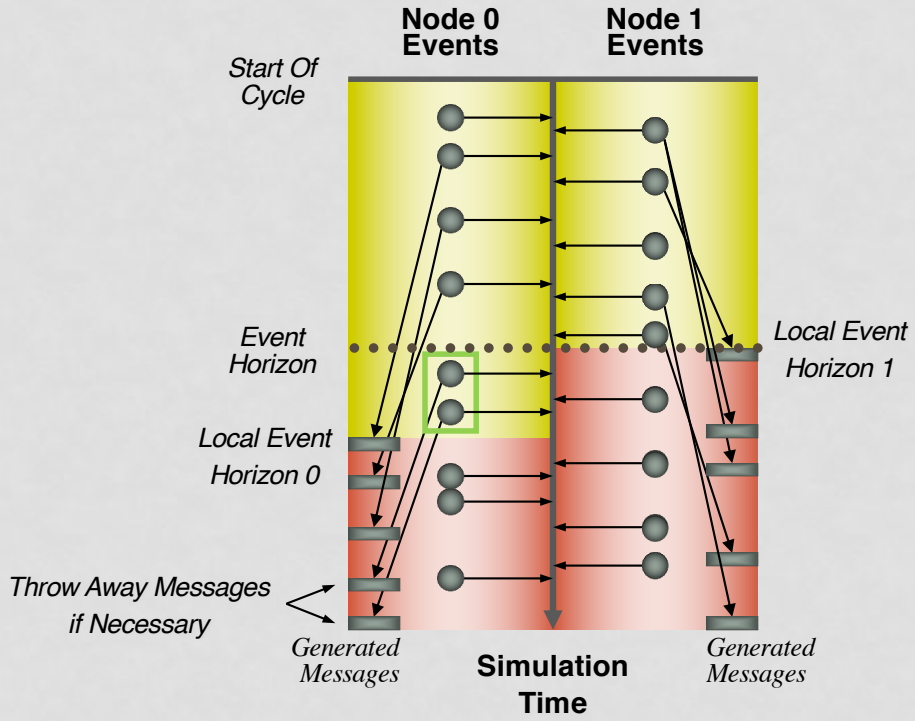
TIME WARP INSTABILITIES



THE EVENT HORIZON



BREATHING TIME BUCKETS

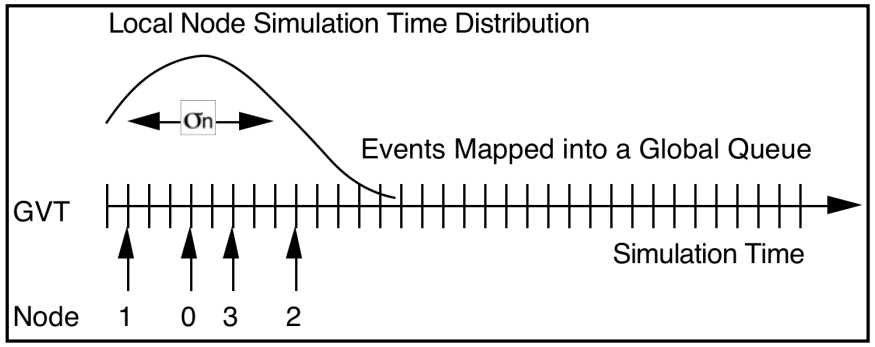


Additional Optimizations

- Asynchronous broadcasts
- Direct insertion of local events

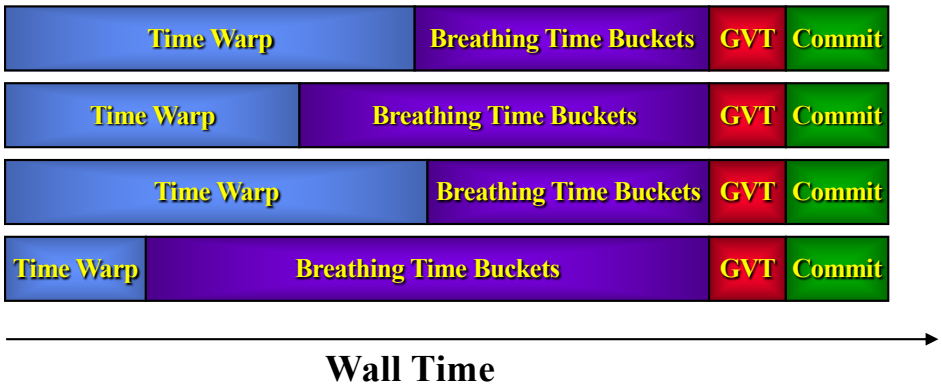
BREATHING TIME WARP

- Opposite problems when comparing Breathing Time Buckets and Time Warp
- Imagine mapping events into a global event queue
- Events processed by runaway nodes have good chance of being rolled back
- Should hold back messages from runaway nodes



BREATHING TIME WARP PROCESSING CYCLE

- Example with four nodes
  - Time Warp: Messages released as events are processed
  - Breathing Time Buckets: Messages held back
  - GVT: Flushes messages out of network while processing events
  - Commit: Releases event horizon messages and commits events



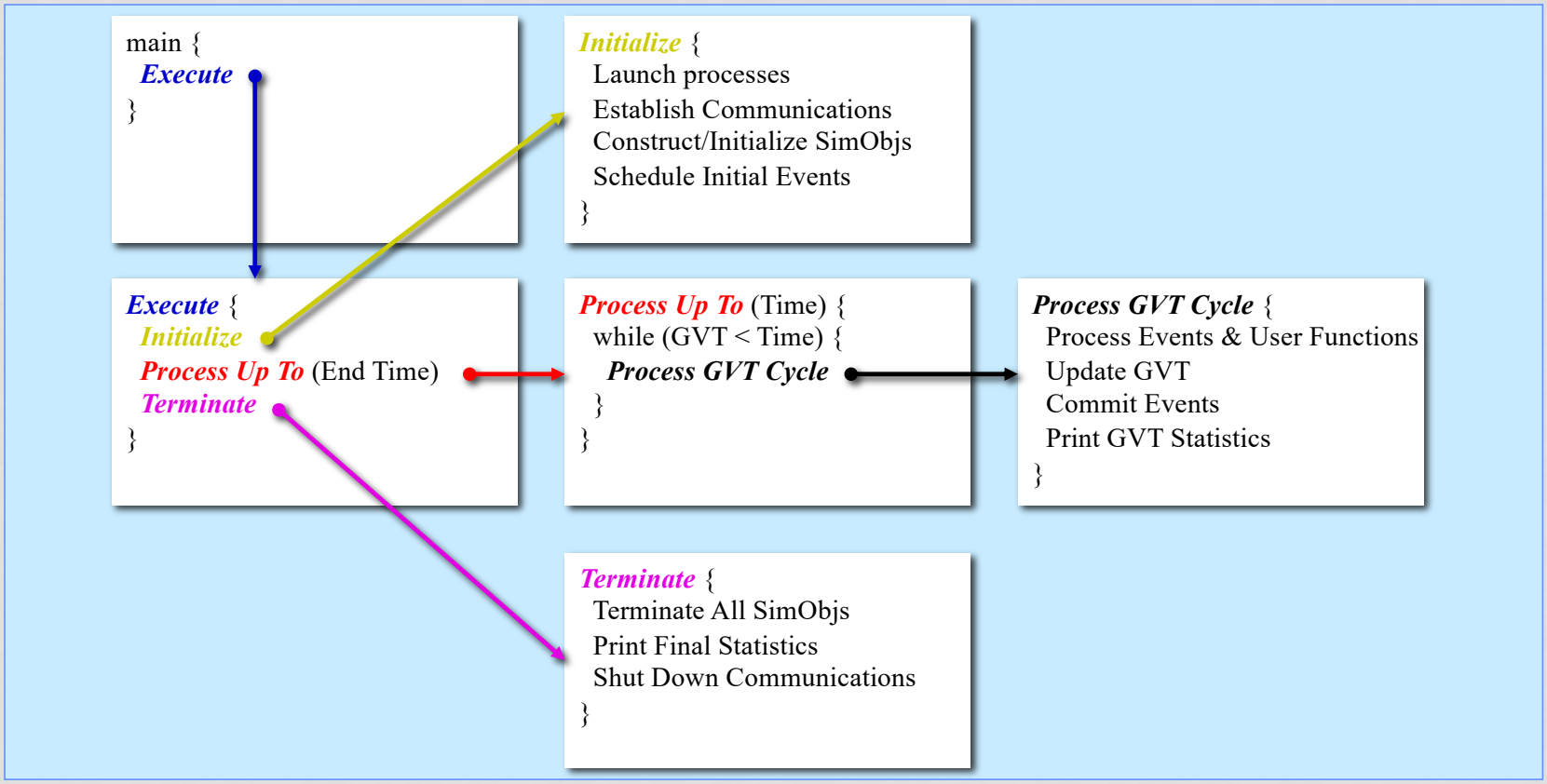
ABSTRACT TIME REPRESENTATION

- Abstract representation of logical time uses 8 tie-breaking fields to guarantee unique time tags
  - double Time — Simulated physical time of the event
  - int Priority1 — First user settable priority field
  - int Priority2 — Second user settable priority field
  - int Counter — Event counter of the scheduling SimObj
  - int UniqueId — Globally unique Id of the scheduling SimObj
  - int AuxCounter — Only set during multi-event transactions
  - int AuxUniqueId — Only set during multi-event transactions
  - double SuperCounter — Unique counter for event instances (for bookkeeping)
- Guaranteed logical times
  - The OpenUTF automatically increments the SimObj event Counter to guarantee that each SimObj schedules its events with unique time tags
    - Note, Counter may "jump" to ensure that events have increasing time tags
    - SimObj Counter = max(SimObj Counter, Event Counter) + 1
  - The OpenUTF automatically stores the UniqueId of the SimObj in event time tags to guarantee that events scheduled by different SimObjs are unique

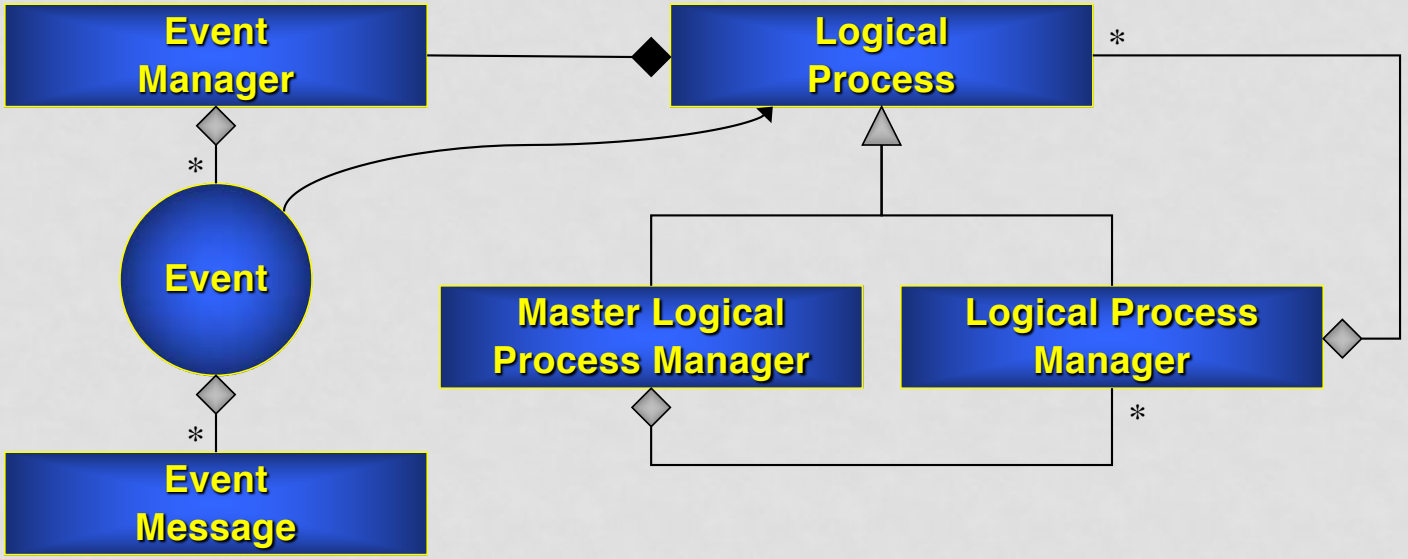
EVENT MANAGEMENT

- Standardized processing cycle interfaces to support any time management algorithm
  - Uses virtual functions on scheduler to specialize processing steps
  - Supports reentrant applications (e.g., HPC-RTI, graphical interfaces, etc.)
- Highly optimized internal algorithms for managing events
  - Optimized and flexible event queue infrastructure
  - Native support for sequential, conservative, and optimistic processing
  - Internal usage of free lists to reduce memory allocation overheads
  - Optimized memory management with high speed communications
- Statistics gathering and debug support
  - Rollback and rollforward application testing
  - Automatic statistics gathering (live critical path analysis, message statistics, event processing and rollbacks, memory usage, profiling, etc.)
  - Merged trace file generation for debugging parallel simulations that can be tailored to include rollback information, performance data, and user output

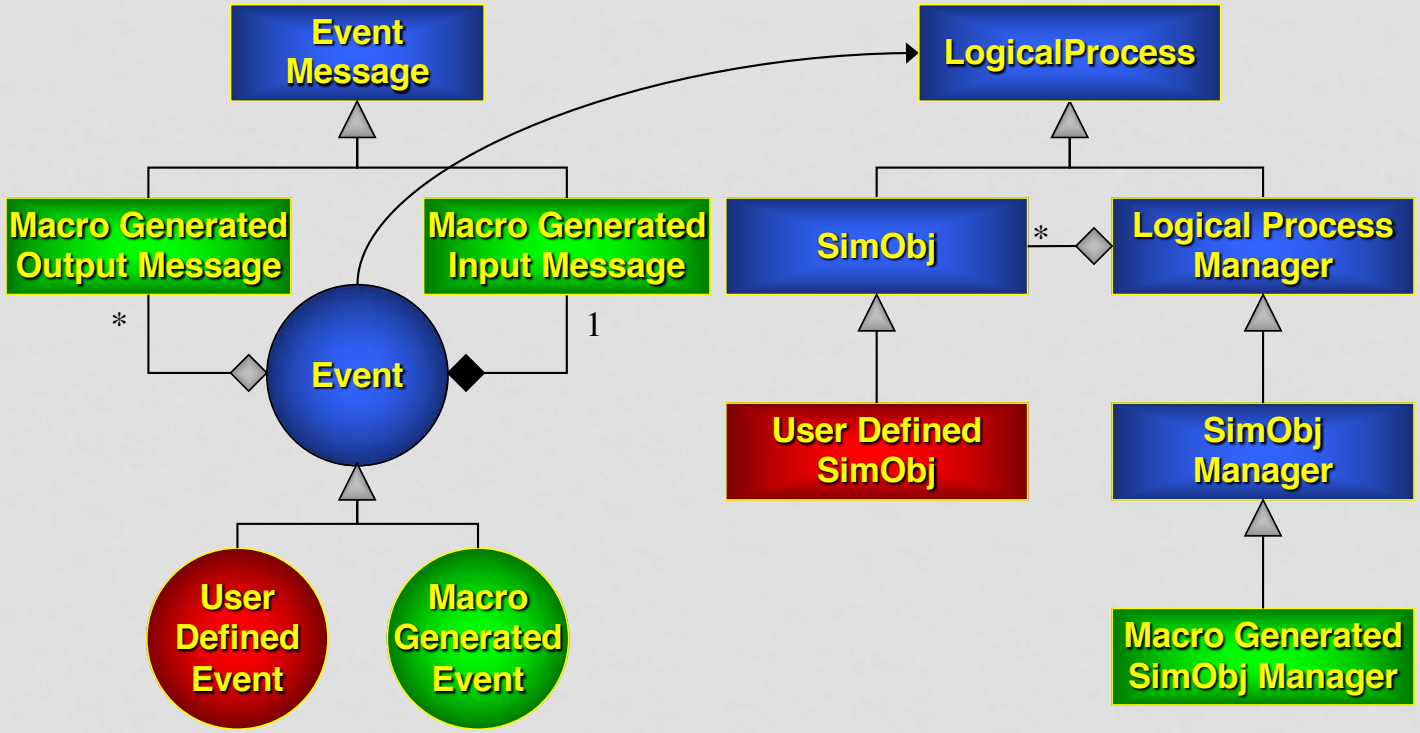
MAIN EVENT PROCESSING LOOP



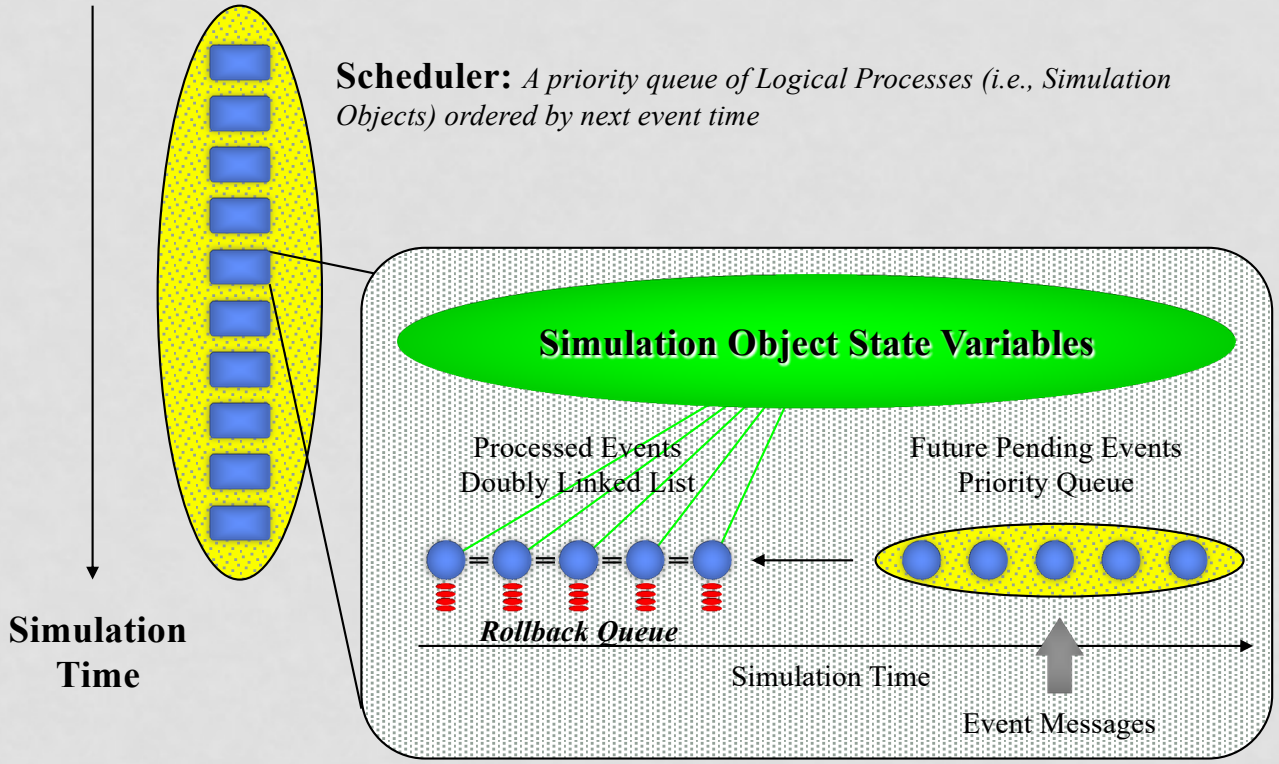
BASIC EVENT MANAGEMENT CLASSES



EVENT MANAGEMENT CLASS RELATIONSHIPS



LOCAL EVENT MANAGEMENT



ROLLBACK MANAGER AND ROLLBACK ITEMS

- Rollback Manager
  - Manages list of rollbackable items that were created as rollbackable operations are performed
  - Each event provides a rollback manager
    - Global pointer is set before the event is processed
    - Rollbacks are performed in reverse order to undo operations
- Rollback Items
  - Each rollbackable operation generates a Rollback Item that is managed by the Rollback Manager
    - Rollback utilities include (1) native data types, (2) memory operations, (3) container classes, (4) strings, and (5) various misc. operations
  - Rollback Items inherit from the base class to provide four virtual functions
    - Rollback, Rollforward, Commit, Uncommit

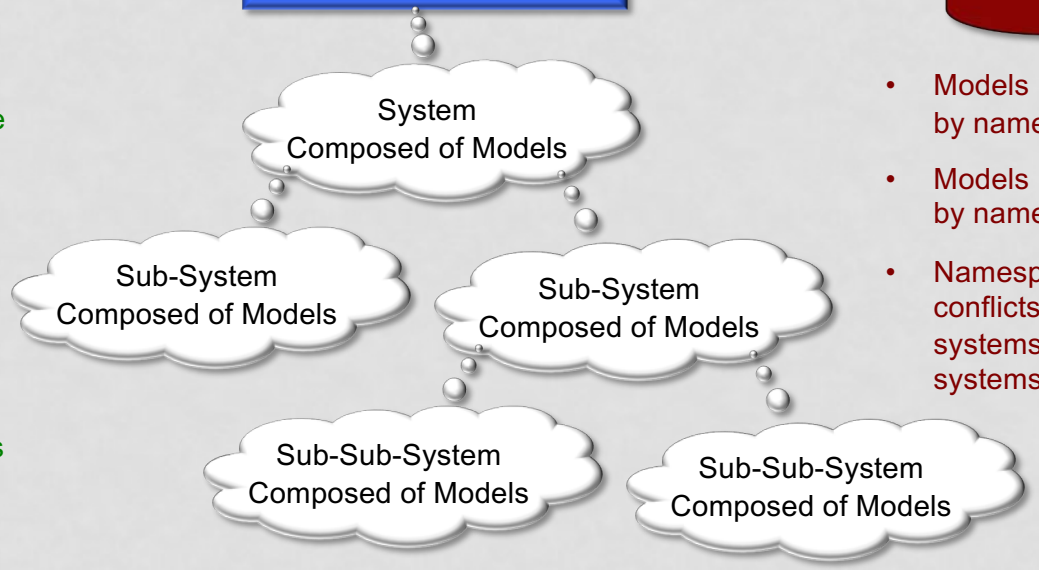
MODELING FRAMEWORK

MODELING FRAMEWORK HIGHLIGHTS

- Composable Modeling
  - Composites, which contain hierarchical systems of models, are automatically constructed/initialized and distributed across nodes by the WarpIV Kernel at startup using a scenario file
  - Models within a composite are co-located (i.e., reside on the same node and evolve in time together as a unit), which means they can share data and involve each other's methods. However, models never know about the existence of other models, which means special abstractions are required (data registry and abstract interfaces)
  - Models residing within different composites interact and share data through publish and subscribe services that still maintain the idea that models know nothing about the existence of other models.
- Event types
  - The WarpIV Kernel Modeling Framework supports a wide variety of event types between co-located and remote models, but the most common type of event is a local event, which is often made to be re-entrant using "Process Model" constructs that support WAIT, WAIT_FOR, Resources, Interrupts, and ASKS
  - Events are scheduled with code-generated interfaces and processed as methods invoked on objects with up to 20 arguments and optional variable-length data

ABSTRACTION OF INTERFACES AND DATA SHARING WITHIN COMPOSITES

- Models dynamically register methods to handle abstract interfaces
- Abstract Interface Manager calls registered methods when invoked or scheduled
- Abstract Interface Manager can also "wire" specific inputs and outputs between models as specified within systems



- Models register data by name and tag
- Models retrieve data by name and tag
- Namespaces resolve conflicts within systems and sub-systems

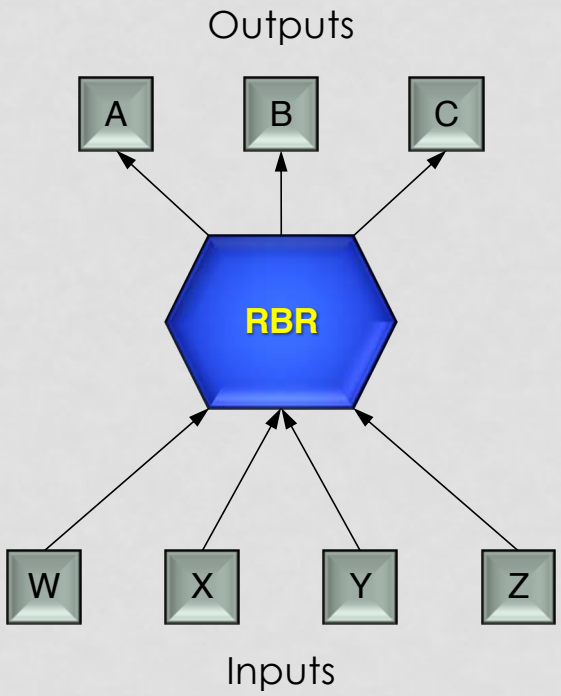
Standards-based ***Publish and Subscribe Data Distribution*** techniques are required for sharing data and supporting interactions between Composites within the OpenUTF and legacy systems

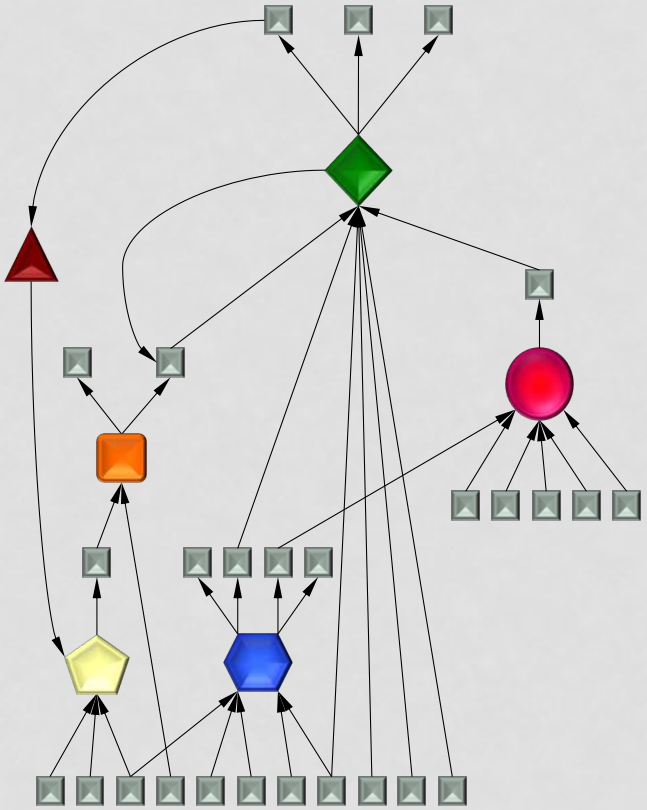
COGNITIVE MODELING

- Cognitive Thought Triggering
  - Thoughts are represented as methods on objects that are triggered whenever any of its input stimulus variables are modified
  - Thoughts can modify output variables that are stimulus to other thoughts, which triggers cognitive thinking in a cascading manner
  - Feedback loops are supported, which model thought refinement
  - Thought termination occurs when there are no more thoughts, or after a maximum number of thoughts have been triggered
- Goals and Task Behaviors
  - Goals are dynamically prioritized and mapped to Task Machines that dynamically carry out a series of tasks that are necessary to accomplish the goal (e.g., GPS driving directions from point A to point B when road conditions might change)

COGNITIVE MODELING & THOUGHT TRIGGERING

Thoughts are methods that are triggered when inputs (stimulus) change, which can produce outputs (stimulus) that trigger other thoughts

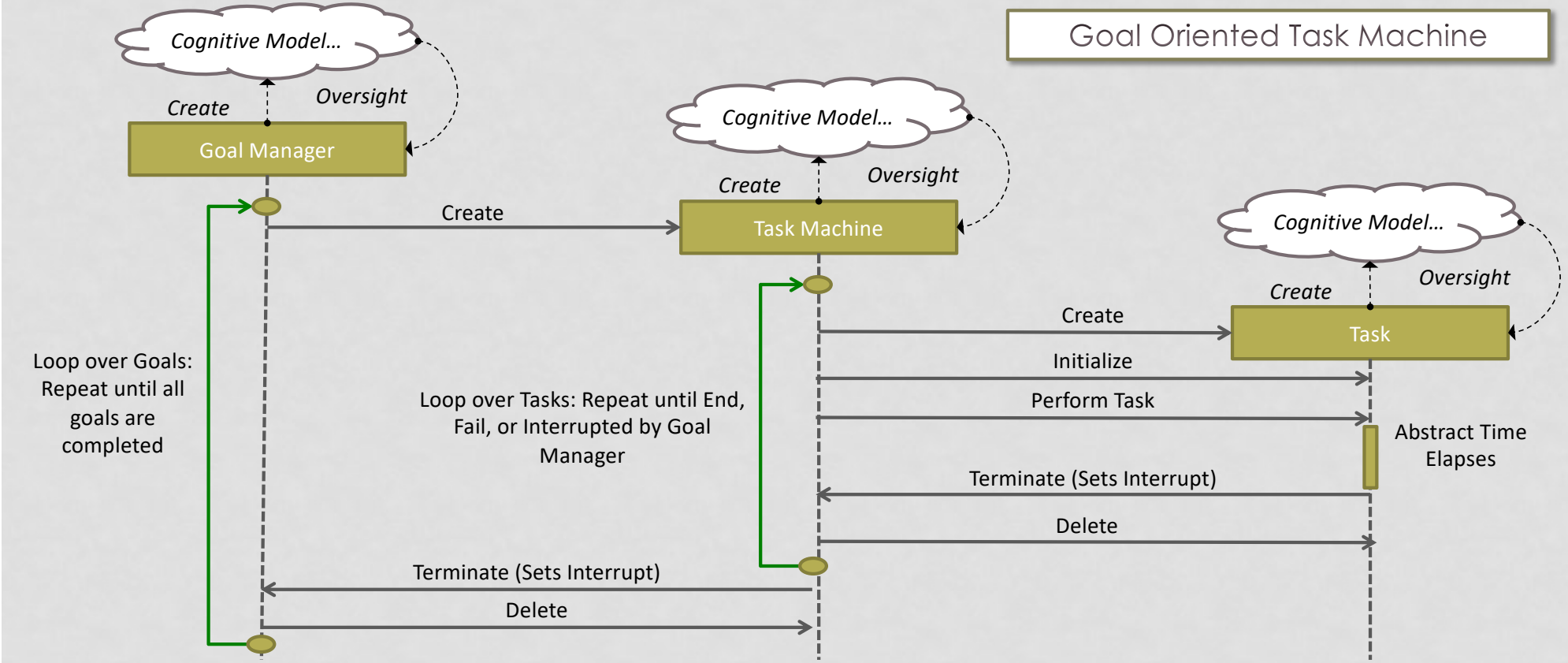




Legend

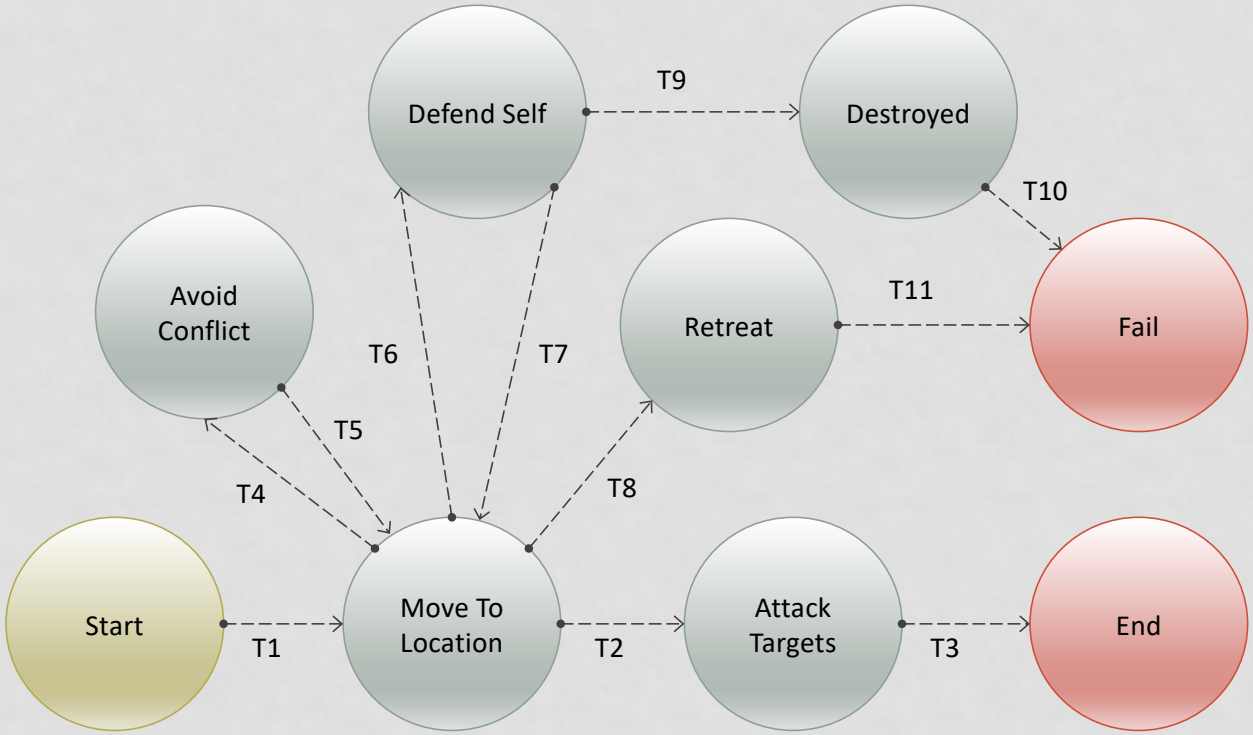
- Emotion Based Reasoning
- Training Based Reasoning
- Rule Based Reasoning
- Predictive Based Reasoning
- Optimized Based Reasoning
- Game Theory (Nash Equilibrium)
- Stimulus (Short Term Memory)

GOAL-ORIENTED TASK MACHINE



TASK MACHINE TRIES TO GO FROM THE START TASK TO THE END TASK USING THE SHORTEST PATH

Tasks are completely encapsulated from one another…



LARGE ASSORTMENT OF SOFTWARE UTILITIES

LARGE ASSORTMENT OF SOFTWARE UTILITIES

- Software Utilities
  - Math utilities for vector operations, matrices, statistical algebra, curve fitting, closest approach calculations, multivariable optimization, unit conversions, constants, rotations, etc.
  - Motion algorithms and coordinate system transformations (ECI, ECR, Round Earth, WGS84)
  - Container classes and other data structures
  - Dynamic memory management with object factories
  - Interpreter of string-based mathematical expressions
  - Data parsers (Config file format, Run Time Class format, XML, CSV, etc.)
  - Random number generation (based on the Mother of All Random Number Generators)
  - Tracking algorithms (various Kalman Filters, chi-squared track fusion, etc.)
  - Error handling
  - Signal processing (FFT, WAV file processing for audio, etc.)
  - Basic Simulation Engine for supporting embedded simulation or representing test federates
  - Dynamic routing algorithms
  - Network-safe data types
  - Artificial Intelligence algorithms (e.g., neural networks, genetic algorithms, etc.)

FEDERATIONS

FEDERATIONS AND THE HIGH-LEVEL ARCHITECTURE (HLA)

- HLA is a 25-year-old standard that is heavily used by the Department of Defense M&S community
  - A federation is group of simulations (called federates) that execute (and evolve time) together as a distributed enterprise
  - Supports distributed objects and interactions (using publish and subscribe services) between the federates that can operate in both real-time and logical-time modes
  - Defined as (1) a Run Time Infrastructure (RTI) that specifies interfaces, (2) a set of rules that all federates must follow, and (3) an object model that all federates agree upon for representing (a) objects and their attributes and (b) interactions and their parameters
- The WarpIV Kernel Provides 2 RTIs (with a 3rd RTI in the future) that are designed for high performance computing in parallel and distributed environments
  - High-Performance Computing Run Time Infrastructure (HPC-RTI) and the LrcProxy Framework are designed to support large-scale federations executing in parallel
  - Important use case is to self-federate a legacy simulation to obtain scalable performance
  - External Modeling Framework (EMF) is designed to support remote federates that might dynamically join/resign while the simulation is running
  - Each of these RTIs are provided with a unified middleware framework that greatly simplifies the federation process while maintaining scalable performance

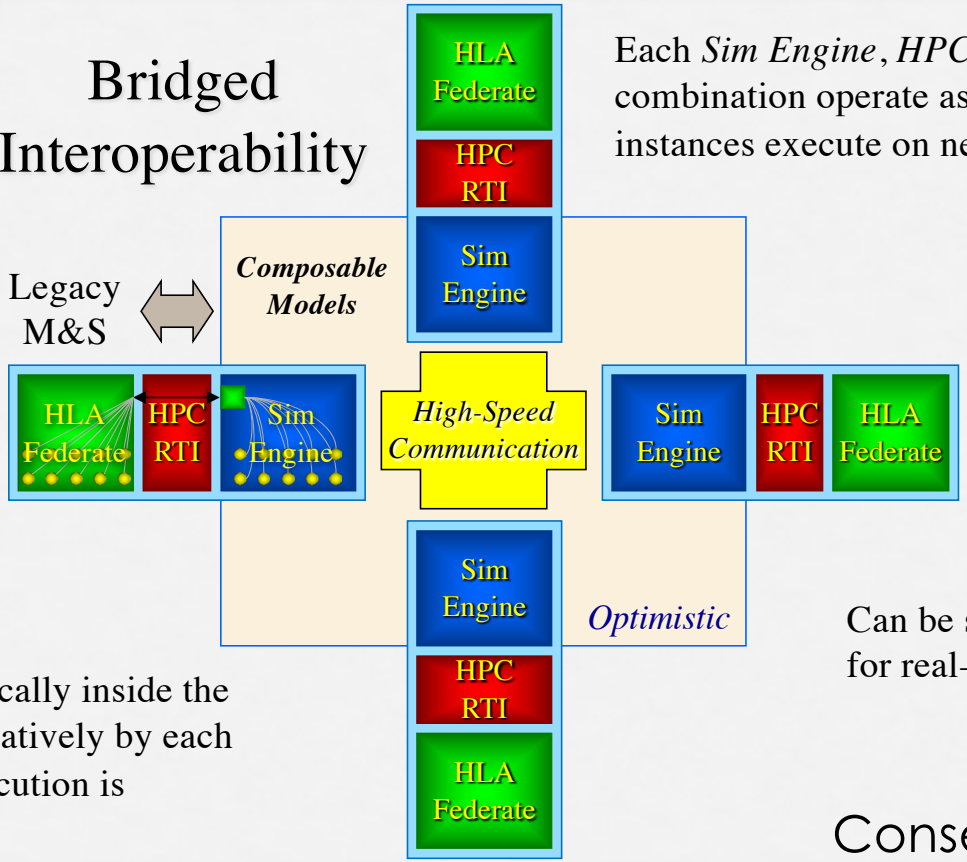
HPC-RTI FOR CONSTRUCTIVE FEDERATIONS WITH COMPOSABLE MODELS & GUARANTEED REPEATABILITY

Unification…

- Composable Models
- Distributed Federations

Could map other distributed technical framework interfaces to the HPC-RTI

Bridged Interoperability



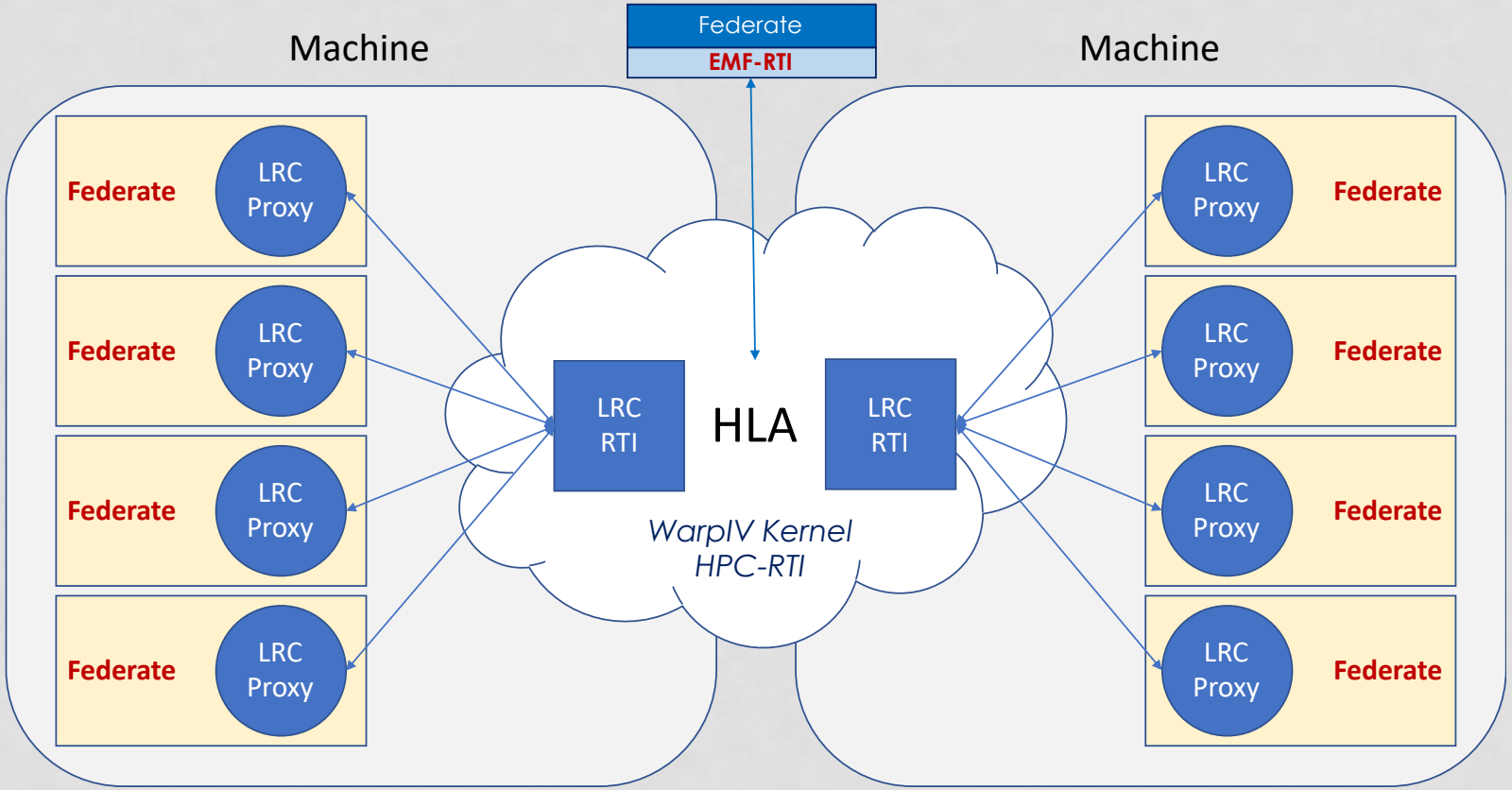
Each *Sim Engine*, *HPC-RTI*, and *HLA Federate* combination operate as a single process. Multiple instances execute on networks of multicore machines.

Can be synced (or scaled) to wall clock for real-time operation

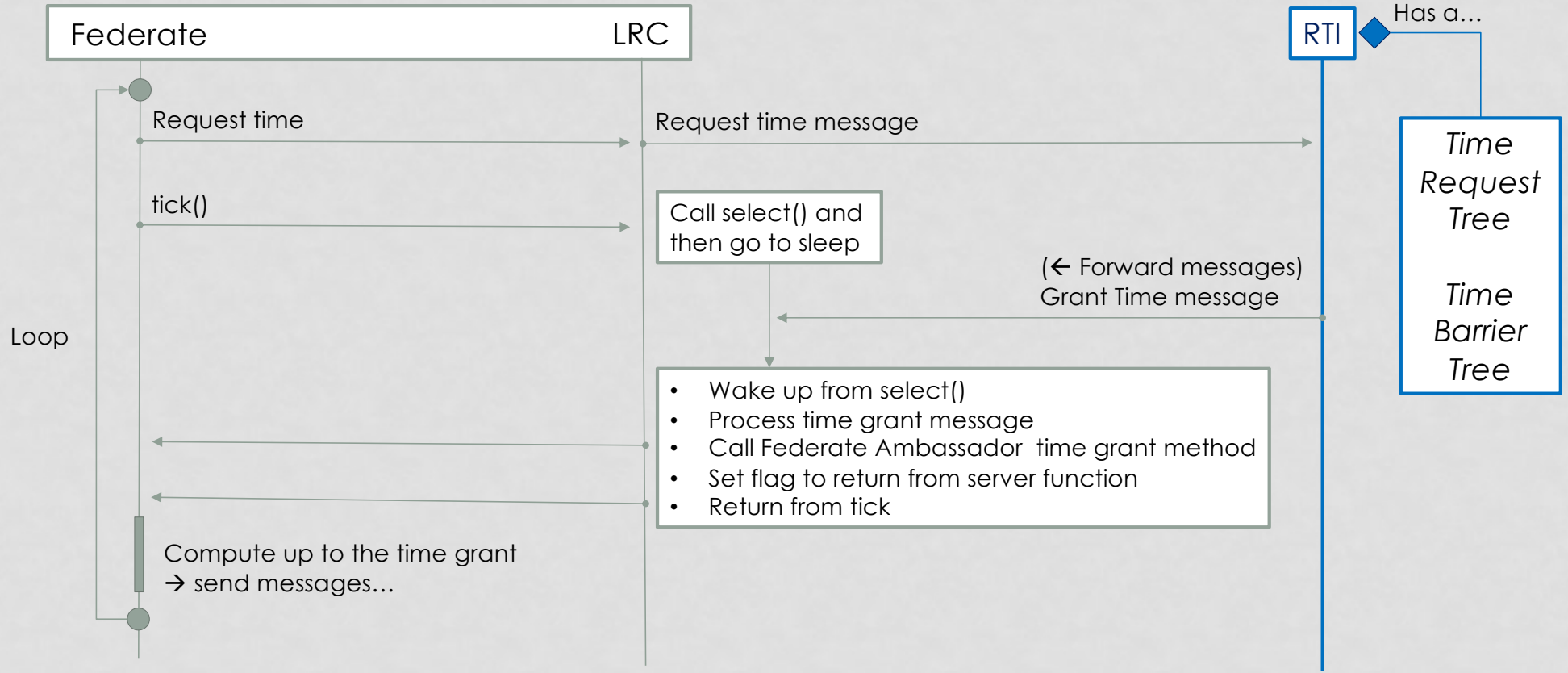
Events are processed optimistically inside the simulation engine, but conservatively by each HLA Federate. Federation execution is guaranteed to be repeatable.

Conservative

LOCAL RTI COMPONENT (LRC) PROXY COLLAPSES THE COMPUTATIONAL FOOTPRINT OF A FEDERATION



COORDINATING TIME REQUESTS AND GRANTS WITH MESSAGE SENDING AND RECEIVING



BIT COMPRESSED PARALLEL AND DISTRIBUTED DATA LOGGING

SUMMARY OF THE BIT COMPRESSED DATA LOGGER

- WarpIV bit-compressed data logger
  - Very easy to use interface
  - More than 100 data types and bit resolutions supported
  - Methods for outputting messages and errors
  - Advanced compression algorithms (including prediction techniques)
  - Filtering based on string expressions that are parsed and evaluated at run time
  - Compression factor of more than 23 observed in missile defense analysis data
  - Tools for printing log files, comparing log files, and printing trace files
  - General-purpose framework for reading and analyzing log files with unstructured data
  - Log files can be password encrypted for improved Information Assurance security
  - Integrated with the Analyzer and Visualizer tools
  - Foundation for cataloging log files with metadata for big data data mining
  - Already in heavy use (repeatability testing in the HPC-RTI, federate behavior data collection for performance predictions, summary of federation performance, optimistic parallel data logging in MDEM, trace file generation for debugging, federate data logging, etc.)

FUNDAMENTALS OF DATA LOGGING

- Logged data is always packed in time-stamped records
  - Time stamps can be logical time or real time
  - A new record is automatically created when time advances and data is logged
  - Log files are always stored as bit-compressed records sorted in ascending time order
  - Logged data can be unstructured and/or structured and occur without any prescribed order
  - Interfaces are provided to read back and access the data
- Each logged data item is a highly compressed triplet
  - <symbol name, data type, value>
  - Symbol names are bit coded and maintained in a dictionary along with their data type
  - Values are bit compressed
- Writing log files
  - Each node, federate, or application can write their own log files
  - Logged data is automatically time-merged and written by one of the federates (logical time)
  - Logged data is sent to a server where it is automatically time-merged and written (real time)

DATA COMPRESSION OF FUNDAMENTAL TYPES

- Integer data types (32-bit integer, 16-bit short, 8-bit char) uses Huffman coding scheme to code the number of bits required to store values
  - Favors small values – e.g., integer value 0 requires only 2 bits of storage
- Floating point data types (64-bit double, 32-bit float) uses a variety of compression techniques to reduce the number of bits
  - Customized floating-point formats for float16, float20, float24, float28, float32, float36
  - Ratios (values between 0 and 1) supported as ratio8, ratio16, ratio24, and ratio32
  - Angles (values between 0 and π, or 0 and 2π) supported as angle8, angle16, angle24, angle32
- Boolean stored as a single bit for true or false
- Strings are always bit coded
  - Huffman (we know all the symbols in advance
  - Dynamically generated Bit Codes discovered during run time
  - Coded Strings convert to bit-coded format strings followed by integer or floating-point values
- Time
  - Years, Weeks, Days, Hours, Minutes, Seconds, Milliseconds, Microseconds, Nanoseconds)

DATA COMPRESSION OF FUNDAMENTAL TYPES CONTINUED

- Unit vectors stored as 3 sign bits and two ratios of selectable resolution
- General vectors stored as a unit vector with selectable resolution and an amplitude with a matching resolution
- Buffers with compression
  - Repeated bytes containing all 0 or 1 bits
  - Repeated bits
  - 4-bit Huffman coding
- Continuous integer and double arrays at various bit resolutions using difference, linear, or quadratic predictions to reduce the number of bits required to store values (like algorithms used for compressing audio)
- Text using a combination of key words and Huffman coding of ASCII characters
- Simulation time and Wall Time (note, WarpIV provides a synchronized wall clock service to obtain absolute times across a parallel and distributed execution)
- Stream I/O operators for logging software-generated messages
- Named Collections (critical for supporting analysis)

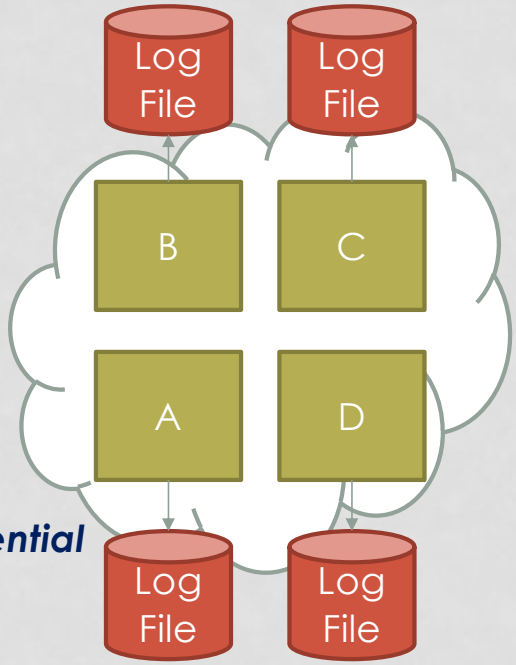
NAMED COLLECTIONS

- A collection of data values that are logged together in time as a unit
  - Named collections have a name for the collection that identifies its type, an integer ID to record who is writing the data, and then a set of data values <symbol name, type, value>
  - Because each named collection type always writes the same set of data, it is not necessary to store their symbol names after the first time.
  - Internal data structures maintain a history of the previous 3 named collections for each ID to support native, difference, linear, and quadratic predictions
    - This provides huge data size reductions for values that are generally continuous (e.g., [X, Y,X] values representing a trajectory, inflight time with fixed steps, numbers of assets that changes slowly, etc.)
  - Named collections have been designed to work with a string-based mathematical interpreter that parses expressions (that evaluate to true or false) using data in the named collection and attributes of published/subscribed objects to determine if the data should be logged (i.e., a smart filter to eliminate logging unnecessary data)
  - Named collections have been designed (but not yet integrated) to work with the Analyzer GUI to perform mathematical analysis of logged data (time plots, scatter plots, histograms, curve fitting, filtering, forming new values, etc.)

EXAMPLE OF A NAMED COLLECTION

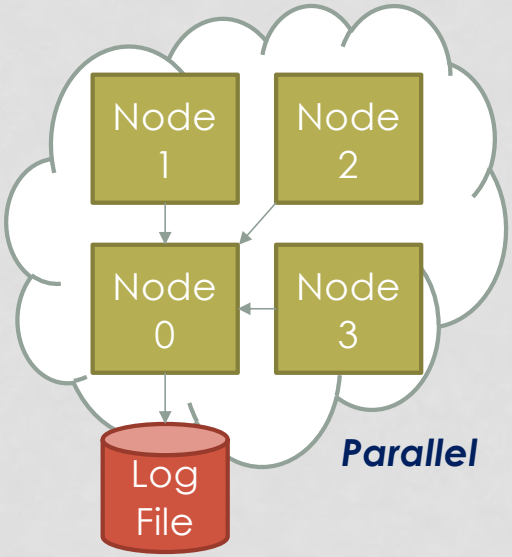
```
WpNamedCollectionRecord *record;

// Create a Named Collect Record named "Moon"

record = APP_DATA_LOGGER.CreateNamedCollectionRecord(
  "Moon",
  RELATIVE_ENUM_ID
);

// Set various values in the record

  record->SetFloat24("X", x[0], WP_BIT_COMP_LOG_CUBIC);
  record->SetFloat24("Y", x[1], WP_BIT_COMP_LOG_ CUBIC);
  record->SetFloat24("Lat", pos[0], WP_BIT_COMP_LOG_DIFF);
  record->SetFloat24("Lon", pos[1], WP_BIT_COMP_LOG_QUADRATIC);
  record->SetFloat24("Alt", pos[2], WP_BIT_COMP_LOG_DIFF);
  record->SetFloat24("Speed", speed, WP_BIT_COMP_LOG_DIFF);
  record->SetFloat24("R", r, WP_BIT_COMP_LOG_DIFF);

// Log the named collection record.

  APP_DATA_LOGGER.LogNamedCollectionRecord(record);
```

DATA LOGGING FILE WRITING MODES



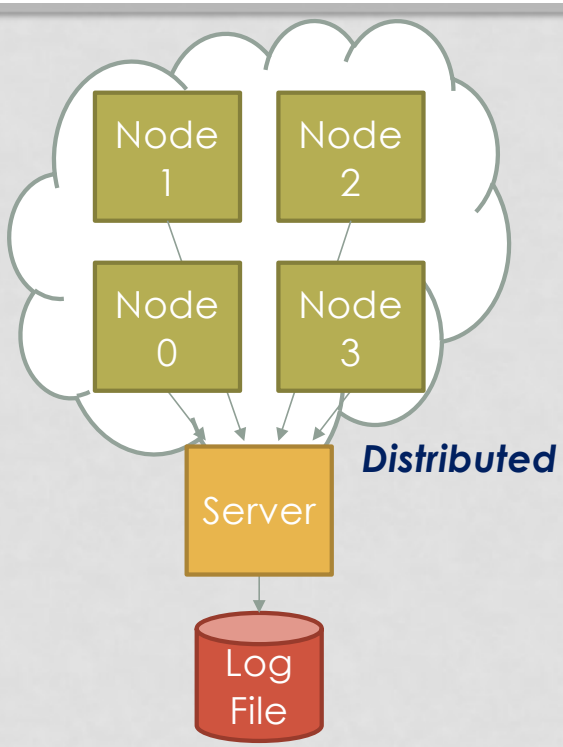
Sequential

Each federate independently writes its own log file



Parallel

The RTI hosting a federate merges and saves a single log file using high speed communications to internally transmit/receive records



Distributed

A network server merges and saves a single log file

BIT COMPRESSED PARALLEL AND DISTRIBUTED DATA LOGGING

OUTPUT STATISTICS

```
Event processing statistics by event type
------------------------------------------
  Id   EventName                          Nproc          Ncmtd           Tcpu          Tcmtd           Tmax       <Tcmtd>           Eff
   2   AddFoToSubscribingEntity           15937           9057      0.0927209      0.0545591     5.6001e-05    6.02397e-06      0.588422
   7   AddPublisherToPubSubFoMgr           2013           2009     0.00271401     0.00240401    0.000101001    1.19662e-06      0.885777
   9   AddSubscriberToEntity                 47              9        0.43534       0.069558       0.013913     0.00772867      0.159779
  11   AddSubscriberToPubSubFoMgr          3022           3022     0.00270802     0.00270802     2.0001e-05    8.96103e-07             1
  14   CreateFoOnFoDistributor              186             72      0.0173842     0.00782107    0.000533001    0.000108626      0.449896
  18   DistributeSimpleEvent              76755          68136        4.35311        3.89286    0.000814001    5.71336e-05      0.894271
  33   InitSectorMotion                    1875           1000        0.27303       0.118492    0.000874001    0.000118492      0.433989
  35   LogMoonXY                            123             51     0.00289212     0.00137805     9.6001e-05    2.70206e-05      0.476484
  36   LogSectorXY                       182759          50000        2.20068       0.565441    0.000848001    1.13088e-05      0.256939
  37   MoonMotion                            13              7     0.00254601    0.000978007    0.000317001    0.000139715      0.384133
  67   SectorMotion                      619841         536400        1293.34        1118.59       0.006273     0.00208536      0.864886
  77   SimpleEvent                      9544977        8644190        31.1755        26.0736    0.000958001    3.01632e-06      0.836351
  78   SimpleEventLocalSubscribe           2000           2000       0.004839       0.004839     5.9001e-05     2.4195e-06             1
  80   SimpleEventRemoteSubscribe           112            112    0.000141112    0.000141112     6.001e-06    1.25993e-06             1
  86   TouchEvent                          2187           2000     0.00139419       0.001256     1.7001e-05      6.28e-07      0.900883


    Event processing totals


        Nproc    = 10451847
        Ncmtd    = 9318065
        Tcpu     = 1331.9
        Tcmtd    = 1149.38
        Tmax     = 0.013913
        <Tcmtd>  = 0.00012335
        Eff      = 0.862965


Critical path analysis
----------------------
    Critical path time = 14.9359
    Processing time = 1149.38
    Maximum speedup = 76.9547
    Efficiency = 0.862965
    Min committed processing time per node = 142.676
    Max committed processing time per node = 145.55
    Mean committed processing time per node = 143.673
    Sigma committed processing time per node = 0.86156


> Exiting


real 3m12.755s
user 3m0.835s
sys 0m10.782s
```

PARALLEL SOFTWARE PROFILING

```
WarpIV Software Profile Utility
Time Mode Wall
Key: PercentTime -> Function() (FileName Line) Time [NumCalls, NumCommittedCalls] <AverageTime> (Overhead, PercentOverhead)

--------------------------------------------------------------------------------
 58.87% -> Sector::SectorMotion() (AUTO_Sector.C Line 18) 4104.83 [629115, 522864] <0.00652478> (0.072113, 0.00%)
     Backtrace:
       59.39% WpEvent::ProcessOptCon() (EventManagement/WpEvent.C Line 1077) 4141.6 [1.04077e+07, 9.28432e+06] <0.000397937> (0.434681, 0.00%)
       73.11% WpScheduler::OptimisticallyProcessNextEvent() (EventManagement/WpScheduler.C Line 645) 5097.68 [1.21724e+07, 1.21724e+07] <0.000418791> (0.51705, 0.00%)
       74.57% WpWarpSpeed::ProcessEvents() (TimeManagement/WpWarpSpeed.C Line 46) 5200.03 [17240, 17240] <0.301626> (0.00102735, 0.00%)
       99.94% WpProcessGvtCycle() (Execute/WpExecute.C Line 878) 6968.48 [17240, 17240] <0.404204> (0.000875473, 0.00%)
       99.94% WpProcessUpToLogicalTime() (Execute/WpExecute.C Line 145) 6968.5 [8, 8] <871.063> (1.93119e-05, 0.00%)
       99.94% WpProcessUpTo() (Execute/WpExecute.C Line 203) 6968.5 [8, 8] <871.063> (3.12328e-05, 0.00%)
      100.00% WpExecute() (Execute/WpExecute.C Line 116) 6972.46 [8, 8] <871.558> (4.76837e-05, 0.00%)
     Self/Children:
        5.74% Self 400.394 [629115, 522864] <0.000636441> (0.072113, 0.00%)
       26.87% WpTransObj::GetDynamicPosition() (SomFomTranslation/WpTransObj.C Line 3578) 1874.15 [1.2749e+09, 1.05984e+09] <1.47003e-06> (50.529, 0.72%)
       15.56% WpTransObj::GetDouble() (SomFomTranslation/WpTransObj.C Line 1476) 1085.4 [1.2749e+09, 1.05984e+09] <8.5136e-07> (50.5113, 0.72%)
        5.59% WpFoMgr::GetNextRemoteFo() (DistSimMgt/WpFoMgr.C Line 1128) 389.97 [1.11554e+09, 9.27357e+08] <3.49579e-07> (44.1006, 0.63%)
        3.61% WpFoMgr::GetFirstRemoteFo() (DistSimMgt/WpFoMgr.C Line 1074) 252.317 [3.18726e+08, 2.64959e+08] <7.91641e-07> (12.6787, 0.18%)
        0.70% WpDynFoSpline5Motion::UpdateEndAcceleration() (Utilities/Math/WpDynFoSpline5Motion.C Line 75) 49.1052 [1.27514e+08, 1.06003e+08] <3.85097e-07> (5.13279, 0.07%)
        0.25% WpDynFoElliptical::operator()() (Utilities/Math/WpDynFoElliptical.C Line 694) 17.6185 [3.38511e+07, 2.8479e+07] <5.20469e-07> (19.0627, 0.27%)
        0.20% WpObjectFactory::NewObject() (Utilities/Memory/WpObjectFactory.C Line 496) 14.2003 [3.18491e+07, 3.18491e+07] <4.45863e-07> (2.11485, 0.03%)
        0.13% WpDynFoSpline5Motion::operator()() (Utilities/Math/WpDynFoSpline5Motion.C Line 280) 9.67974 [1.27514e+08, 1.06003e+08] <7.59113e-08> (70.8385, 1.01%)
        0.04% WpDynFoSpline5Motion::Init() (Utilities/Math/WpDynFoSpline5Motion.C Line 35) 3.1931 [3.18491e+07, 2.6477e+07] <1.00257e-07> (19.8736, 0.28%)
        0.03% WpTransObj::SetBuffer() (SomFomTranslation/WpTransObj.C Line 2147) 2.57793 [629115, 522864] <4.0977e-06> (0.180612, 0.00%)
        0.03% WpDynFoSpline5Motion::operator()() (Utilities/Math/WpDynFoSpline5Motion.C Line 245) 2.32138 [3.18491e+07, 2.6477e+07] <7.28869e-08> (1.30032, 0.01%)
        0.02% WpDynFoSpline5Motion::WpDynFoSpline5Motion() (Utilities/Math/WpDynFoSpline5Motion.C Line 20) 1.43394 [3.16893e+07, 2.63448e+07] <4.52502e-08> (1.39845, 0.02%)
        0.01% WpCreateEventMsg() (EventManagement/WpScheduleEvent.C Line 39) 0.848932 [793927, 659221] <1.06928e-06> (0.117202, 0.00%)
        0.00% WpTransObj::SetInt() (SomFomTranslation/WpTransObj.C Line 1332) 0.455884 [138863, 114941] <3.28297e-06> (0.11602, 0.00%)
        0.00% WpScheduleEvent() (EventManagement/WpScheduleEvent.C Line 124) 0.414966 [793927, 659221] <5.22675e-07> (0.0895486, 0.00%)
        0.00% WpTransObj::SetDouble() (SomFomTranslation/WpTransObj.C Line 1428) 0.260437 [138863, 114941] <1.8755e-06> (0.0566089, 0.00%)
        0.00% WpMasterLogicalProcessMgr::GetLogicalProcessMgr() (EventManagement/WpMasterLogicalProcessMgr.C Line 373) 0.208985 [159830, 132203] <1.30754e-06> (0.0432057, 0.00%)
        0.00% WpObjectFactory::GetObjectBytes() (Utilities/Memory/WpObjectFactory.C Line 273) 0.125647 [629115, 522864] <1.9972e-07> (0.927966, 0.01%)
        0.00% WpLogicalProcessMgr_SCATTER::GetObjHandle() (EventManagement/WpLogicalProcessMgr.C Line 493) 0.0601497 [159830, 132203] <3.76335e-07> (0.0216146, 0.00%)
        0.00% WpMasterLogicalProcessMgr::GetAlias() (EventManagement/WpMasterLogicalProcessMgr.C Line 418) 0.053575 [325318, 269504] <1.64685e-07> (0.0953867, 0.00%)
        0.00% WpObjectFactory::GetObjectType() (Utilities/Memory/WpObjectFactory.C Line 260) 0.034981 [325318, 269504] <1.07529e-07> (0.0732112, 0.00%)
        0.00% WpMasterLogicalProcessMgr::GetTypeId() (EventManagement/WpMasterLogicalProcessMgr.C Line 406) 0.00760078 [4982, 4154] <1.52565e-06> (0.00461864, 0.00%)
        0.00% WpTransObj::GetDynamicPosition() (SomFomTranslation/WpTransObj.C Line 3596) 0.00405025 [1000, 1000] <4.05025e-06> (0.000102282, 0.00%)
        0.00% WpObjectFactory::GetObjectTypeId() (Utilities/Memory/WpObjectFactory.C Line 224) 2.00272e-05 [14, 14] <1.43051e-06> (1.83582e-05, 0.00%)
--------------------------------------------------------------------------------
```

CROSS-PLATFORM WIDGET SET FOR BUILDING GUIS, 3D/2D VISUALIZATION, AND ANALYSIS

CROSS-PLATFORM GUI TOOLS

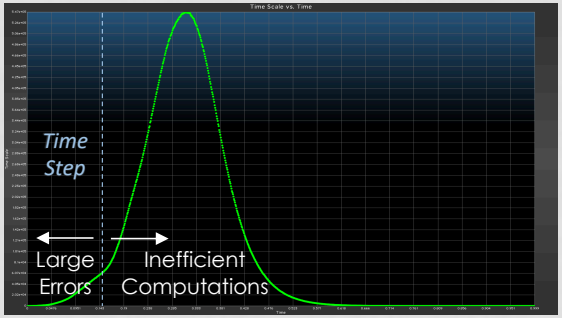
- Main Tools
  - Visualizer provides 2D and 3D playback visualization
  - Analyzer provides mathematical data plotting and analysis
  - Model Editor provides a graphical front end for developing models
  - Scenario Editor with 2D canvas provides a graphical front end for composing systems of models and specifying scenarios
  - Messager GUI provides a time-ordering messaging service for parallel and distributed applications
  - More… (performance report editor/calculator, performance forecaster, missile editor, etc.)

WORK PERFORMED ON THIS PHASE I NASA SBIR EFFORT

DISCRETE-EVENT VS. TIME STEPPING FOR BRUTE FORCE N-BODY GRAVITATIONAL PROBLEMS

$$\vec{F}_{12} = \frac{m_1 m_2 G}{\left|\vec{R}_{12}(t)\right|^3}\vec{R}_{12}(t)$$

$$\delta F_{12} = \frac{m_1 m_2 G}{R_{12}^3}\,|3v_R - v|\,\delta t$$



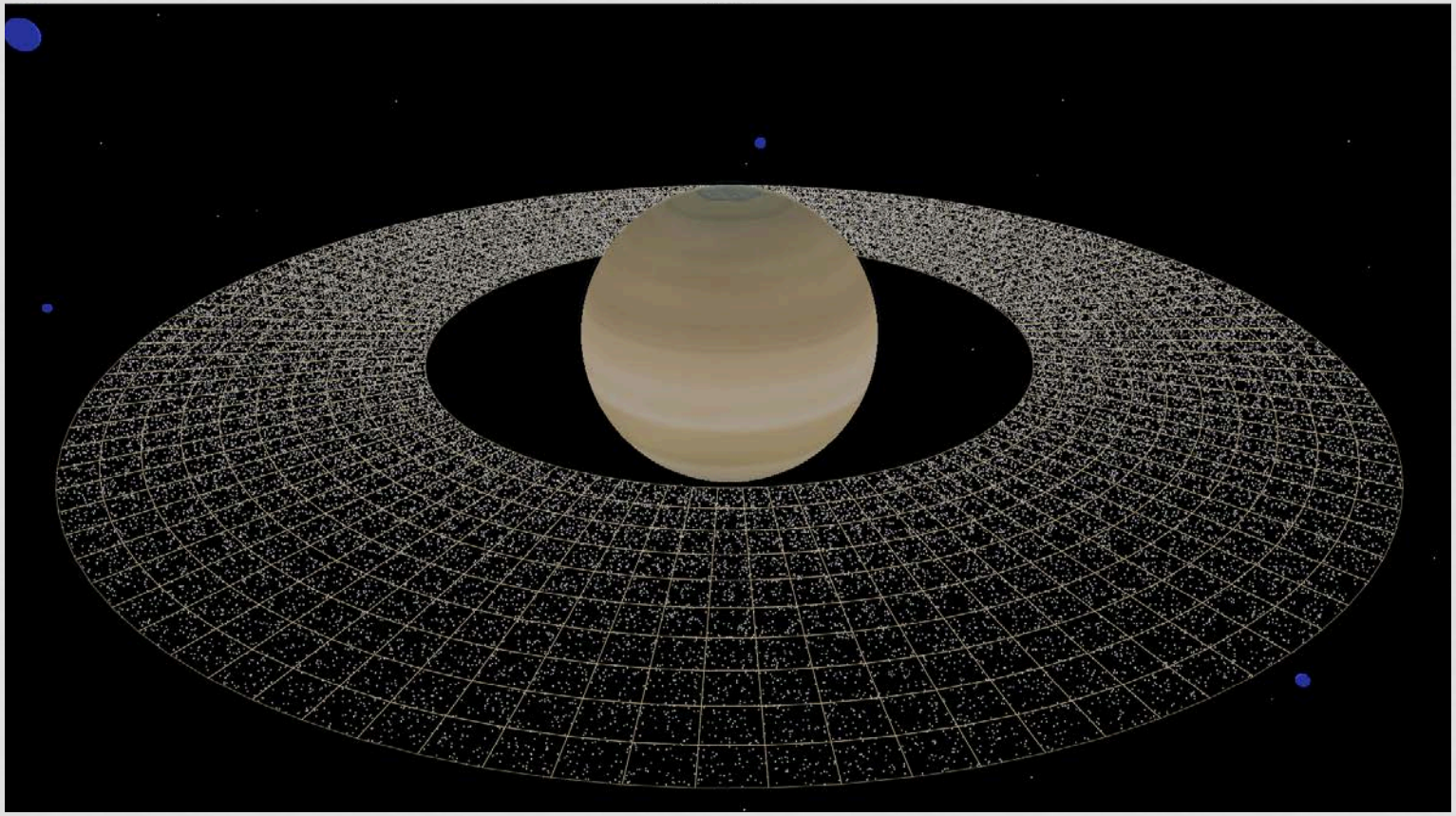
This graph shows the distribution for computational update times between pairs of particles randomly distributed in a circular planetary ring while holding the change in force constant. Note that the horizontal axis representing the natural update time per pair of particles is a log plot. The time stepped approach computes most of the forces far more often than necessary (i.e., computations to the right of the time step), while not accurately supporting the very-tight time scales (i.e., computations to the left of the time step), which is often when the most interesting physics occurs.

Analysis of brute-force N-body gravitational computations for a planetary ring with 1,000 particles shows the following over-computation factors for different time step cutoffs relative to the update time distribution. Time Step cutoffs (i.e., the area under the curve to the left of the time step) at 1%, 10%, 25%, 50%, 75%, 90%, 99%, provide over-computation factors of 16404, 961, 229, 53, 13, 3, 0.2 respectively (note that the 99% cutoff shows extreme under-computing, which would create very large systematic errors making results invalid). So, for 1,000 particles, traditional time stepping at a 1% cutoff to hopefully produce accurate outputs does more than 4 orders of magnitude computations than needed

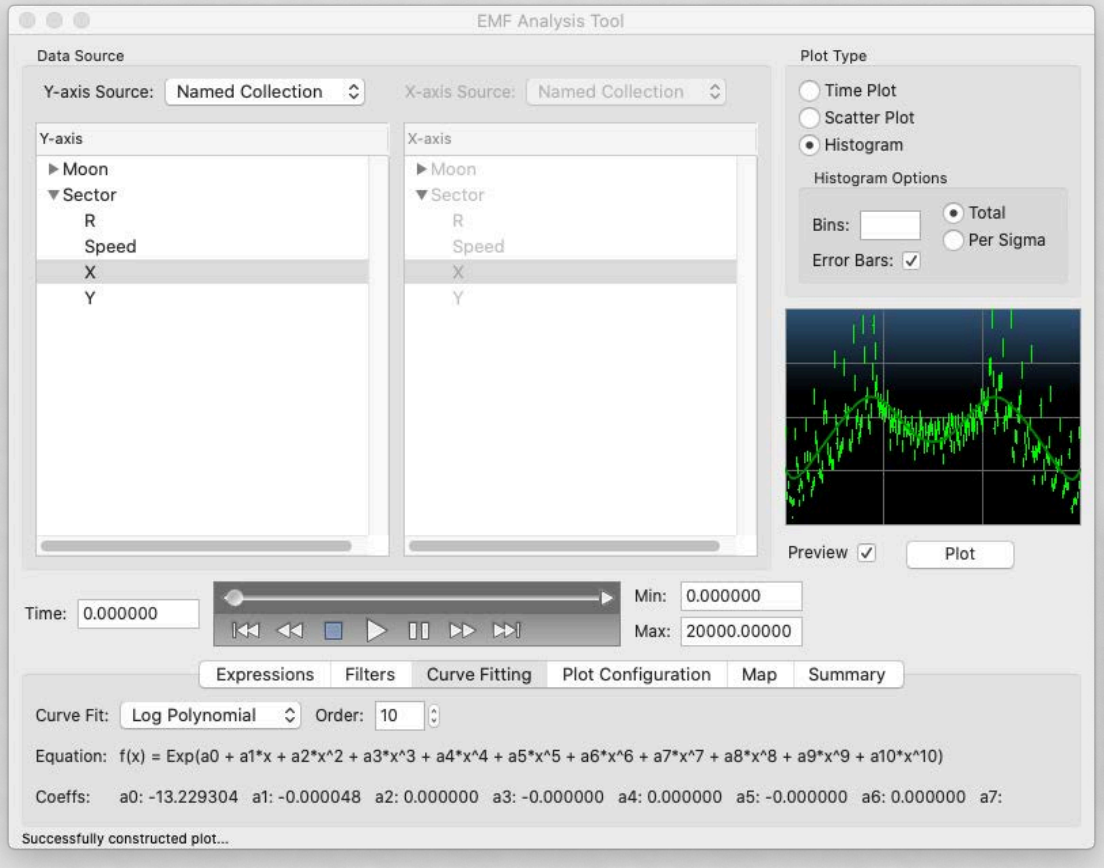
It is better to hold the error tolerance constant and let time scales fluctuate than to hold time scales constant and let error tolerances fluctuate…

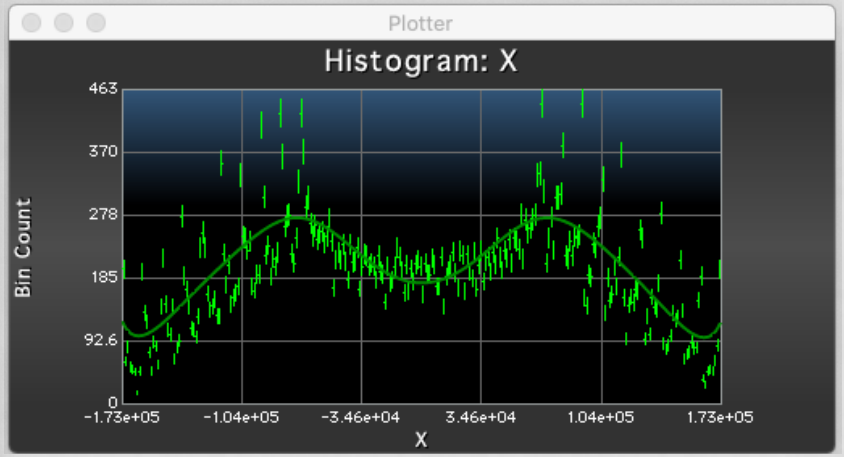
DISCRETE EVENT IS FASTER AND MORE ACCURATE THAN TIME STEPPING

VISUALIZER – PLANETARY RINGS OF SATURN



ANALYZER - SATURN





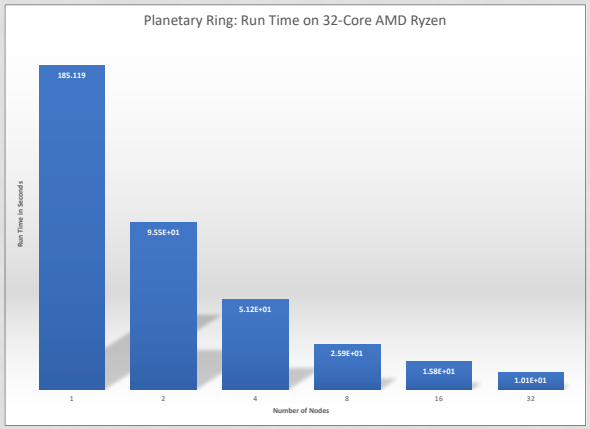
Planet [0, 87, -1, <7, 4, 0>]

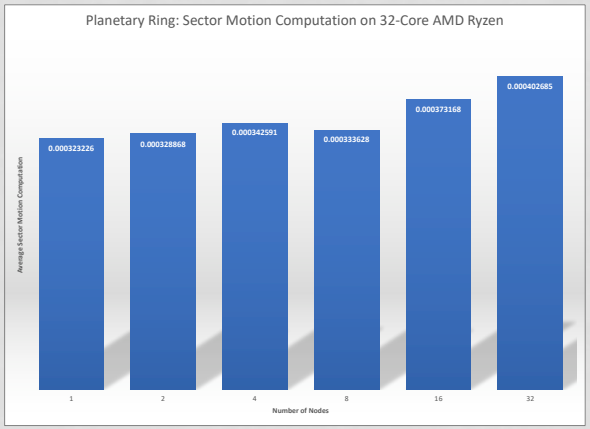
Attribute	Value
Position	{0.000000, 0.000000, 0.000000
Mass	5683400000000000030526144.
Radius	58232.000000
MajorAxis	60268.000000
MinorAxis	54364.000000
MeanRadius	58232.000000

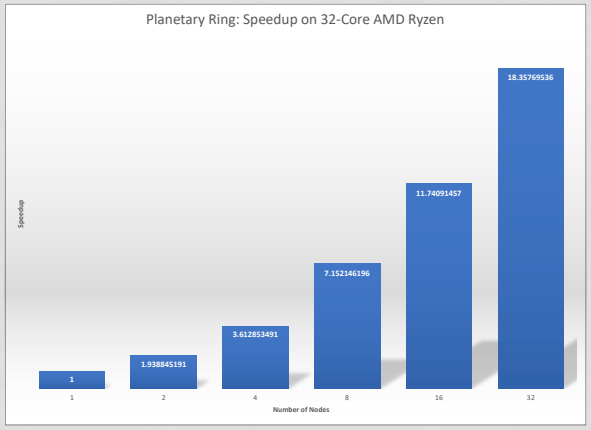
Sector [1, 101, -1, <5, 4, 2>]

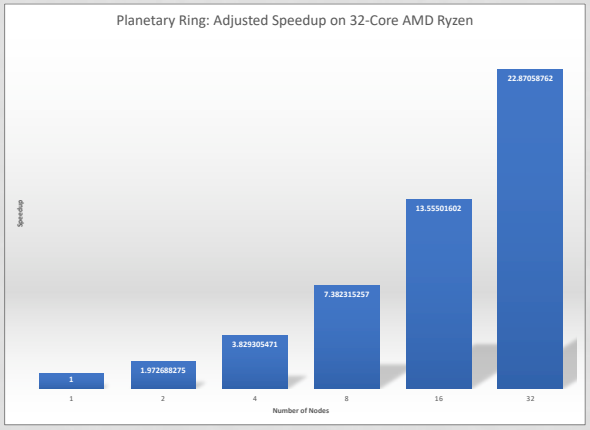
Attribute	Value
Position	{-66908.865345, -22420.8280
Mass	1324400000000000.000000
Radius	
AngularWidth	0.062832
RangeWidth	11407.000000
ParticlesInRing	50000
ParticlesInSect...	43
MassRing	1540000000000000000.0000

PLANETARY RING PERFORMANCE

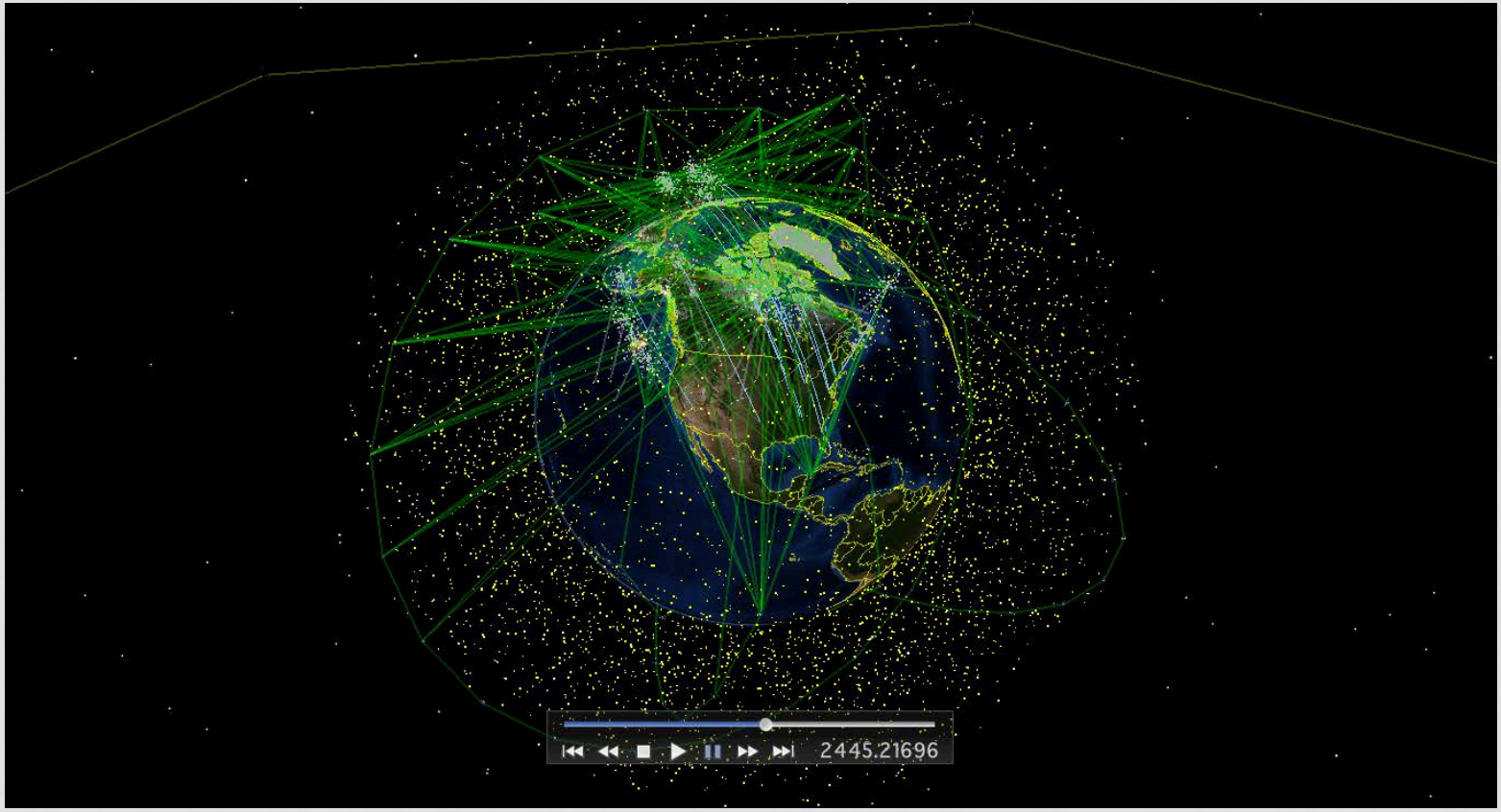




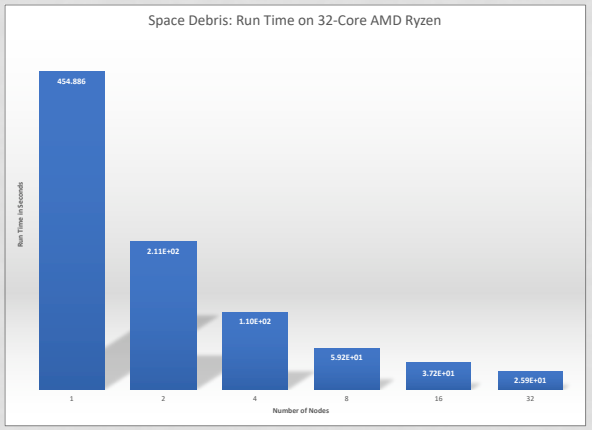


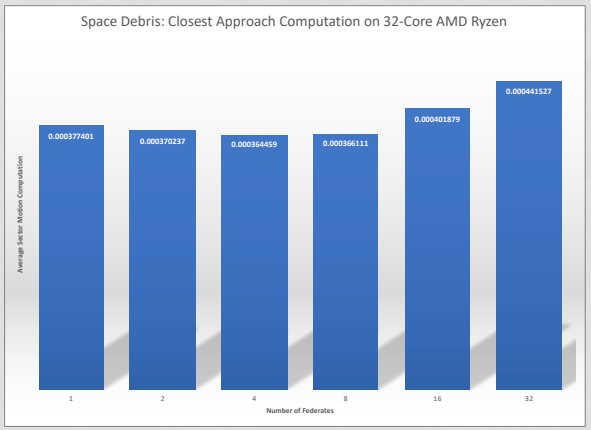


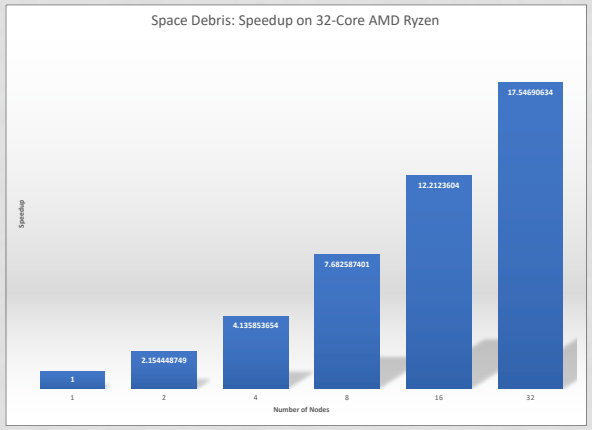
VISUALIZER – MISSILE DEFENSE WITH SPACE DEBRIS

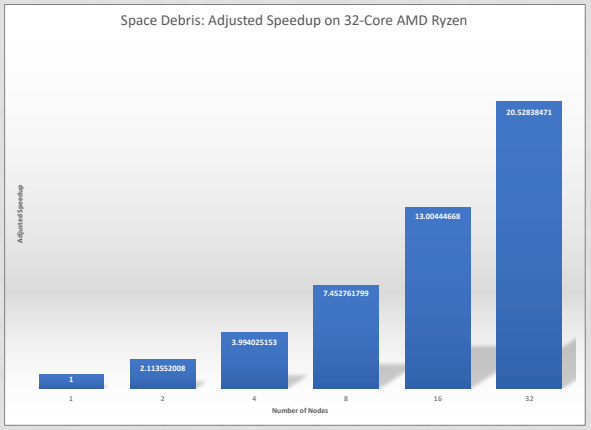


SPACE DEBRIS PERFORMANCE

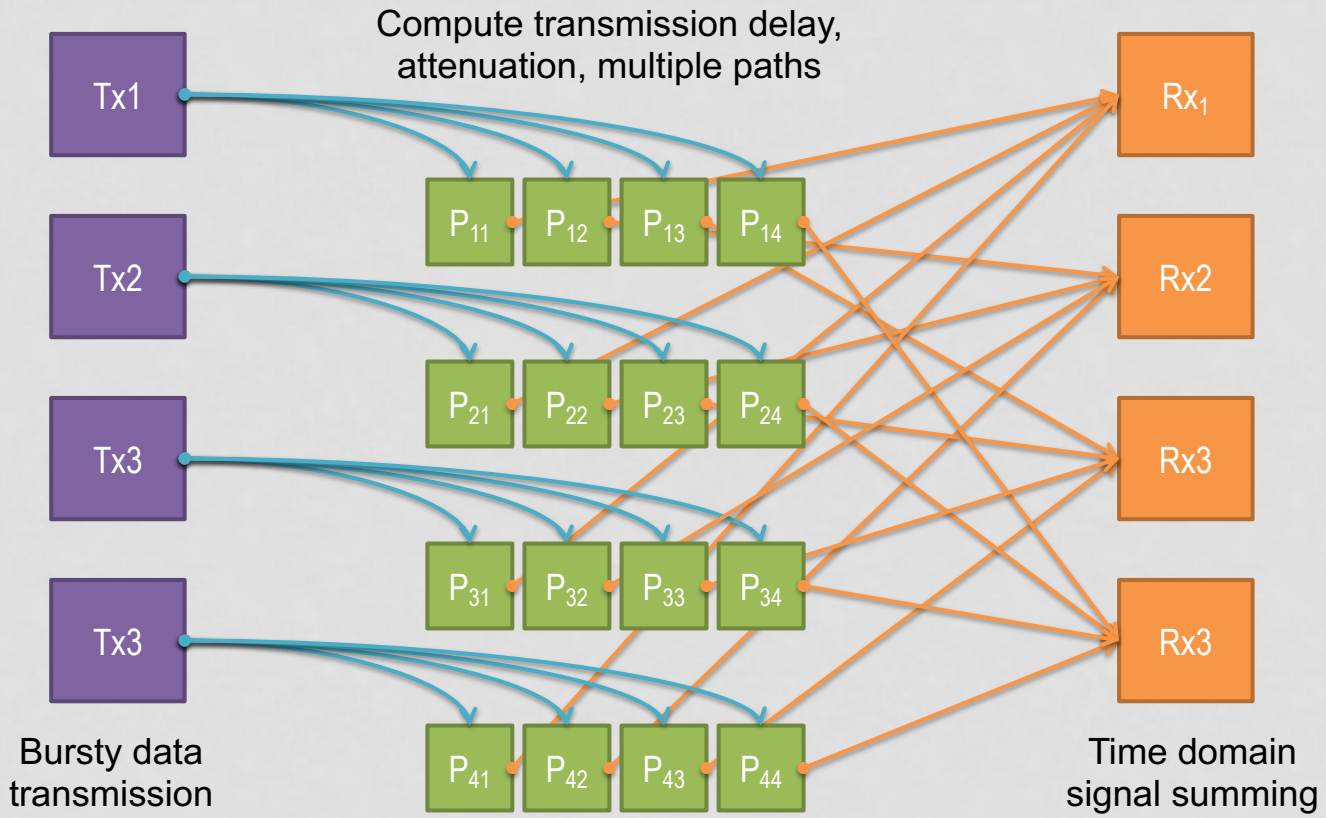




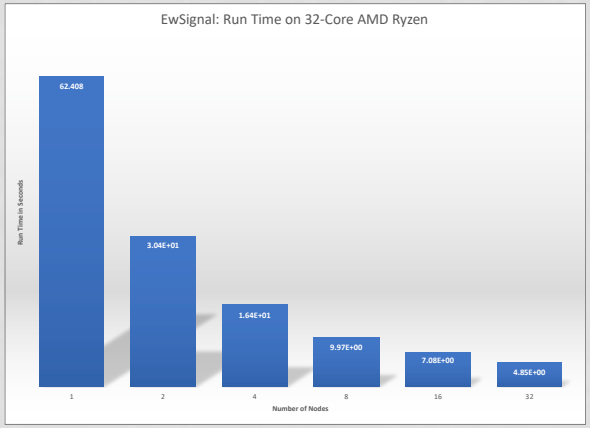


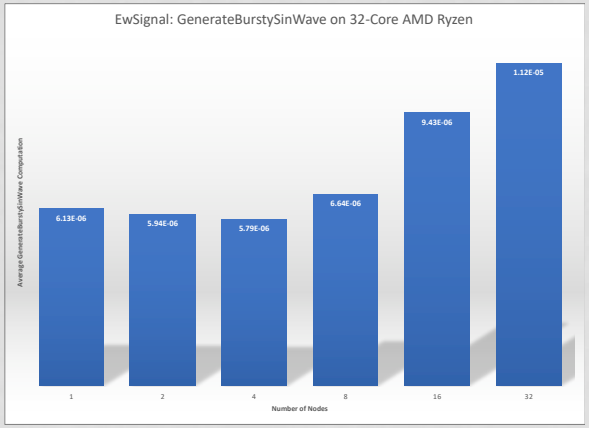


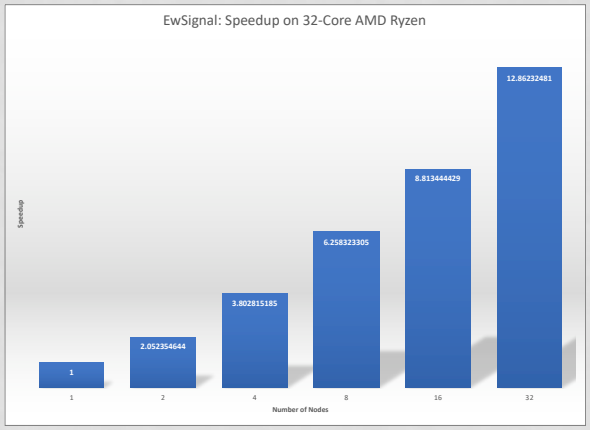
RF SIGNAL PROPAGATION FOR ELECTRONIC WARFARE

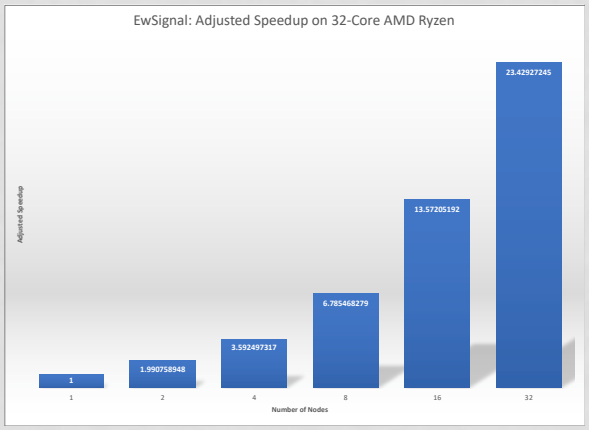


RF SIGNAL PROPAGATION FOR ELECTRONIC WARFARE PERFORMANCE

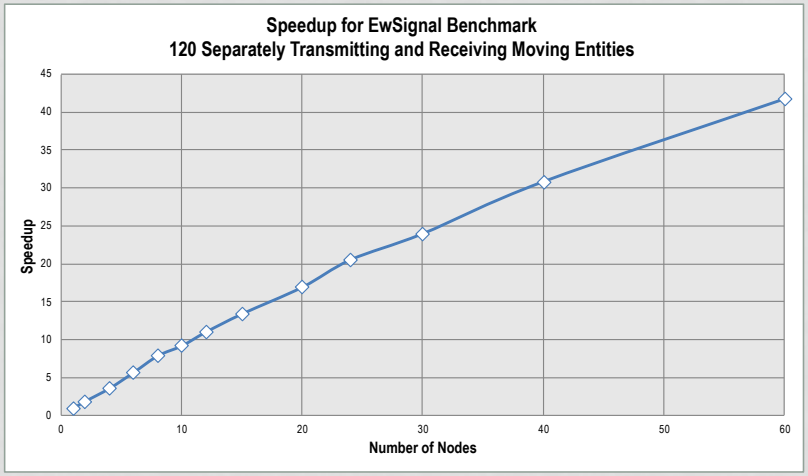








Previous Benchmark Performed at SPAWAR on 60 cores

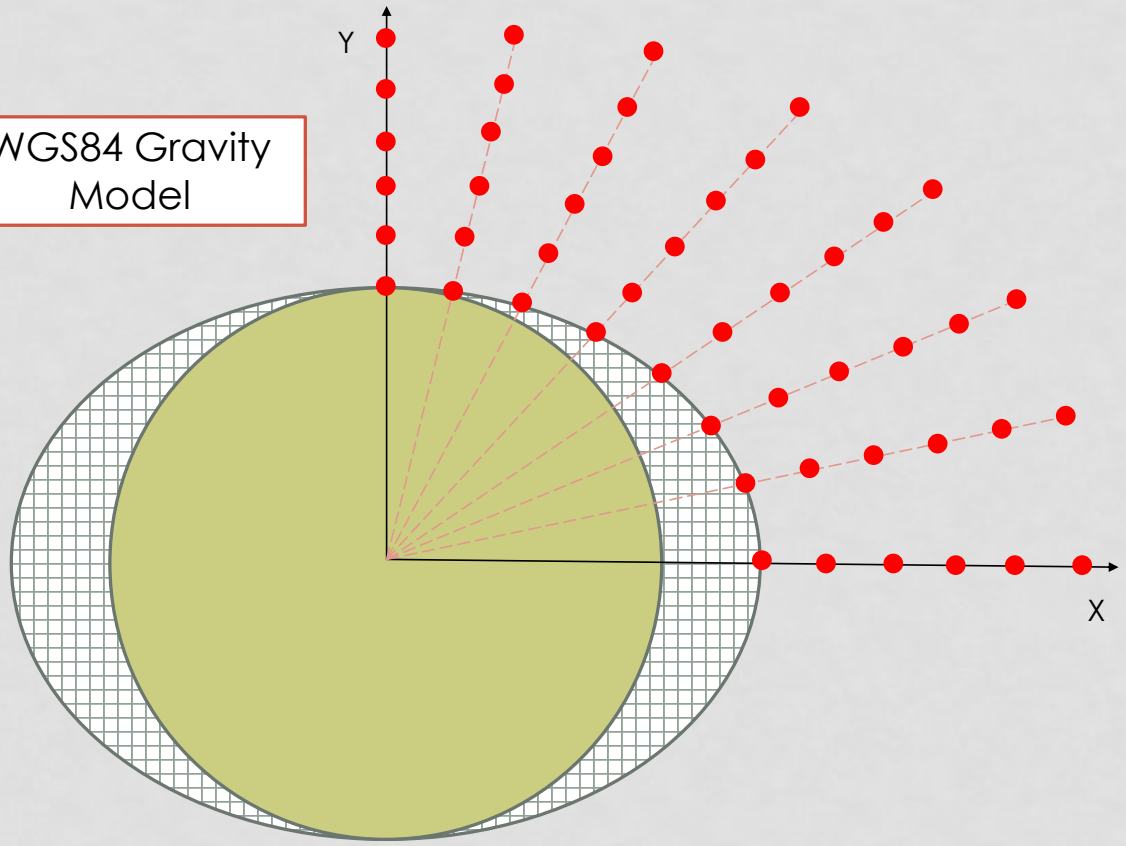


ELLIPSOIDAL GRAVITY MODEL

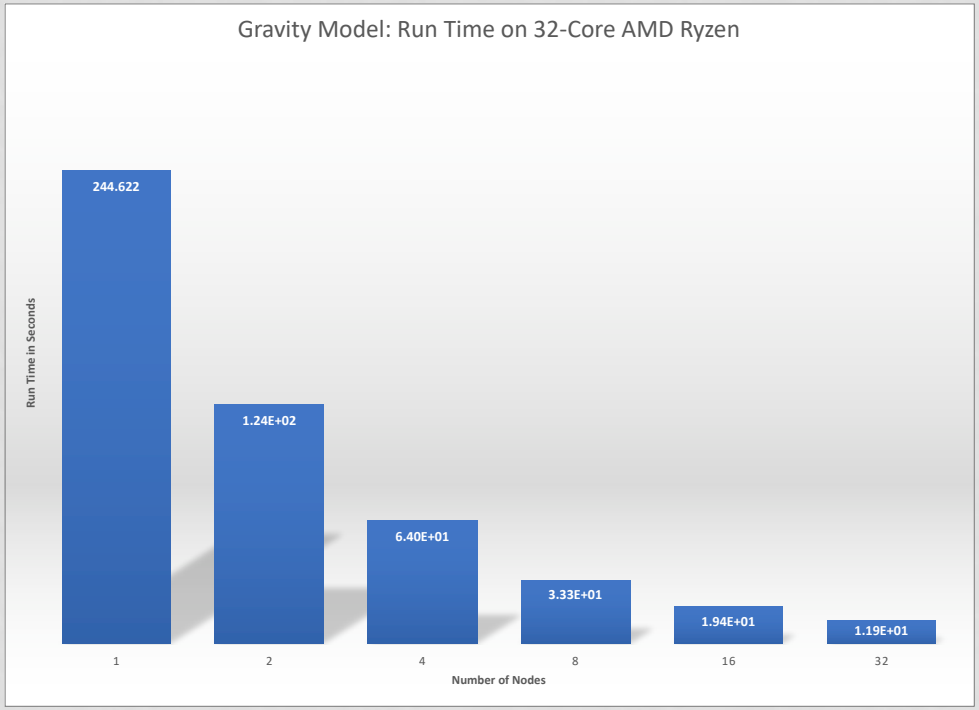
Ellipsoidal gravity model

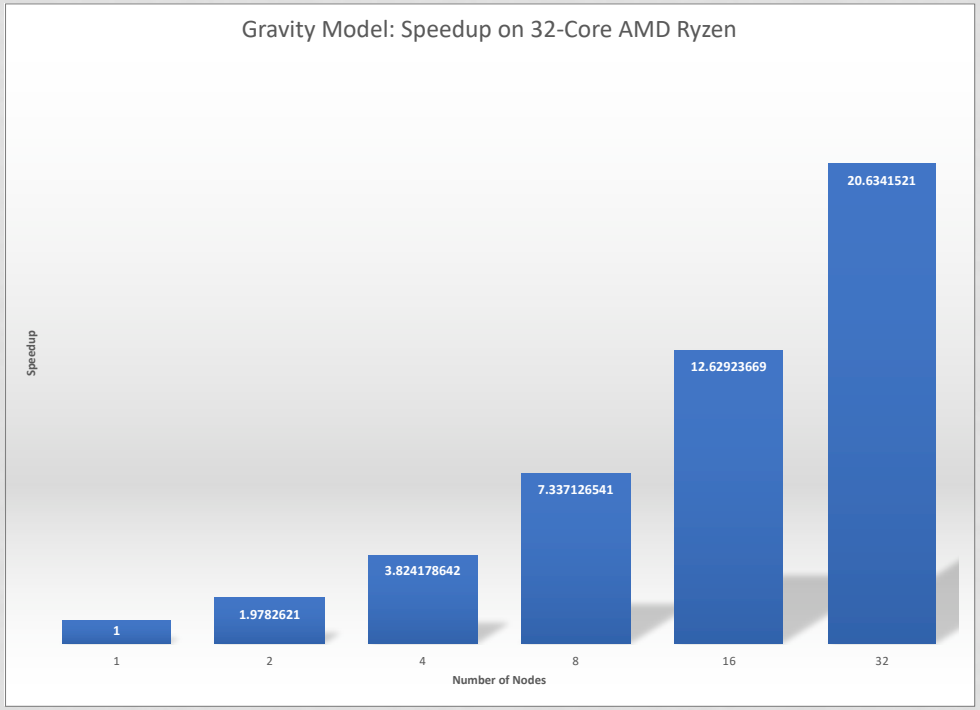
- Subtracts the enclosed spherical mass (representing central gravitational force
- Integrates remaining mass cells at different spatial locations
- Fits a high-order polynomial to determine the additional gravitational contribution to the central force from the spherical mass

WGS84 Gravity Model



ELLIPSOIDAL GRAVITY MODEL PERFORMANCE





FINAL THOUGHTS

- A Phase II effort will bring the WarpIV Kernel to the NASA supercomputing community
  - Training Materials (hands on, pre-recorded videos, future seminars)
  - High Speed Communications
  - Parallel Discrete Event Simulation Engine with advanced Modeling Framework
  - Tons of software utilities (data structures, mathematical tools, memory management, etc.)
  - Debug and Profiling Tools
  - Cross-platform GUIs (Mac, Linux, Windows, UNIX)
  - Parallel and Distributed Bit-Compressed Data Logging
- We are very interested in collaborating with the NASA supercomputing community in areas of Science (scientific/mission models) and Technology (framework) efforts…
  - Install the WarpIV Kernel on NASA HPC Supercomputers
  - Distribute the WarpIV Kernel to users on laptops, desktops, etc.
  - Need endorsements to continue in Phase II

QUESTIONS